Journal of Korea Multimedia Society Vol. 15, No. 1, January 2012(pp. 18-31)

다성음원 기반 QbSH 시스템을 위한 매칭엔진의 설계 및 구현

박성주†, 정광수††

요 약

본 논문은 다성음원에서 추출된 특성정보 기반 QbSH (Query-by-Singing/ Humming) 시스템의 매칭엔진에 대해 제안하였다. 다성음원 기반 QbSH 시스템은 사람의 노래나 허밍에서 추출된 특성정보와 MP3 파일과 같은 다성음원에서 추출된 특성정보를 비교하여, 가장 유사한 음원을 검색하는 시스템이다. 제안된 매칭엔진에는 다성음원에서 특성 추출시 발생하는 오류를 줄이고, 매칭성능을 향상시키기 위해 크로마-스케일 표현기법 (Chroma-Scale Representation), 보상기법 (Compensation) 및 비대칭적 DTW (Asymmetric Dynamic Time Warping) 알고리즘을 적용하였다. 또한 다양한 거리 함수 (Distance Metric)를 적용하여 매칭엔진의 성능향상을 확인하였다. 1,000개의 허밍 질의와 450곡의 다성음원 데이터베이스를 기반으로 제안한 QbSH 시스템의 성능 실험을 수행하다. 성능 평가를 통해 제안한 QbSH 시스템이 MRR (Mean Reciprocal Rank) 0.718의 정확도를 가지는 것으로 확인되었다.

Design and Implementation of Matching Engine for QbSH System Based on Polyphonic Music

Sung-Joo Park†, Kwangsue Chung††

ABSTRACT

This paper proposes a matching engine of query-by-singing/humming (QbSH) system which retrieves the most similar music information by comparing the input data with the extracted feature information from polyphonic music like MP3. The feature sequences transcribed from polyphonic music may have many errors. So, to reduce the influence of errors and improve the performance, the chroma-scale representation, compensation and asymmetric DTW (Dynamic Time Warping) are adopted in the matching engine. The performance of various distance metrics are also investigated in this paper. In our experiment, the proposed QbSH system achieves MRR (Mean Reciprocal Rank) of 0.718 for 1000 singing/humming queries when searching from a database of 450 polyphonic musics.

Key words: QbSH(허밍검색시스템), Matching Engine(매칭엔진), Polyphonic Music(다성음원), Feature Extraction(특성추출)

※ 교신저자(Corresponding Author) : 정광수, 주소 : 서울시 노원구 월계동 447-1 광운대학교 참빛관 809호, 우편번호(139-701), 전화 : 02)940-5134, FAX : 02)919-3218, E-mail : kchung@kw.ac.kr
접수일 : 2011년 7월 14일, 수정일 : 2011년 10월 31일
완료일 : 2011년 11월 22일

† 정회원, 전자부품연구원 디지털미디어연구센터
(E-mail : bpark@keti.re.kr)
†† 정회원, 광운대학교 전자통신공학과
※ 본 연구는 지식경제부 산원원천기술개발사업의 일환으로 수행하였음. [10037244, 차세대 음향 핵심기술 개발 및 산업 활성화 지원센터구축] 지원으로 수행되었음.

1. 서 론

최근 디지털 음원의 이용이 확대됨과 동시에 대용량 저장장치를 지원하는 다양한 미디어 기기가 보급되고 있다. 개인은 미디어 기기에 상당히 많은 수의 음원을 저장하여 이용하고 있으며, 동시에 새로운 음원에 대한 신속한 접근 및 소비에 대한 요구도 점차 강해지고 있는 실정이다. 이러한 음원에 대한 소비 경향의 변화에 따라, 수 만곡을 초과하는 대용량의 음원 데이터에 대한 신속한 색인, 검색 등의 기술에 대한 관심이 증가하고 있다. 음원 검색 (Music Information Retrieval, MIR) 기술에는 허밍 질의 검색 (Query-by-Humming) [1]을 포함하여 음악 장르 분류 (Musical Genre Classification) [2,3], 멜로디 추출 (Melody Extraction) [4,5], 커버-송 인식 (Cover-Song Identification) [6], 템포 추출 (Tempo Estimation) [7] 및 오디오 인식 (Audio Identification) [8] 등의 다양한 분야가 존재한다. 허밍 질의 검색은 사용자의 목소리인 허밍 입력을 기반으로 한다는 측면에서 다른 기술과 구분될 수 있다. 사람의 목소리 입력, 즉 노래와 허밍 기반의 검색 시스템의 개념이 도입된 후, 이 시스템은 MIR 분야에서 주요한 검색 방법으로 자리를 잡았으며, 이는 사용자에게 보다 자유로운 음원 검색 방법을 제공한다는 측면에서 의미가 있다. 노래와 허밍, 두 가지 질의 (Query) 방식을 모두 포함하는 검색 시스템을 QbSH (Query-by-Singing /Humming) 시스템으로 정의하고 있으며, 이러한 시스템은 사용자 입력 수단이 제한적인 모바일 단말에서의 음원 검색에 보다 효율적으로 적용이 가능하다.

지금까지의 대부분의 음원 검색 시스템은 MIDI로 대표되는 단성음원 (Monophonic Music)을 기반으로 데이터베이스를 구성하였다 [9]. 실제 사용자의 목소리 정보가 단성음원의 형태이기 때문에, 검색 성능 측면에서는 단성음원을 기반으로 데이터베이스 및 음원 검색 시스템을 구성하는 것이 유리한 측면이 있다. 그러나 MP3, Wav 파일 등 실제 서비스되고 있는 대부분의 디지털 음원은 목소리, 악기 등의 다양한 음원이 혼재되어 있는 다성음원 (Polyphonic Music)의 형태이다. 따라서 실제 활용할 수 있는 QbSH 시스템의 구현을 위해서는 다성음원을 이용한 시스템 구성이 요구된다.

QbSH 시스템은 사용자의 질의 입력에서의 특성정보 추출하는 부분, 다성음원에서의 특성정보 추출하는 부분 및 추출된 두 특성정보를 비교하는 매칭엔진의 세 부분으로 구성되며, 실제 이 세 부분에 입력되는 모든 데이터에서 엄연히 오류가 존재한다. 즉 QbSH 시스템은 하나의 절대적인 기준을 가지고 이에 대한 유사도를 비교하는 것이 아니라, 오류를 가지고 있는 데이터들 사이에 상대적인 비교를 수행하는 시스템으로 정의할 수 있다. 일반적으로 대표적인 단성음원인 MIDI 파일의 특성정보는 곡의 악보 등 구체적인 정보를 기반으로 생성되기 때문에 상대적으로 정확한 반면, 보컬, 악기 등 다양한 음원이 혼재되어 있는 다성음원에서 주요 특성정보인 멜로디를 추출하는 것은 MIDI만큼 정확하지 않고, 아직 기술적으로 완벽하지 않은 단계로서 추출된 특성정보에는 많은 오류가 포함되어 있다. 또, 단성음원의 형태인 사용자 허밍질의의 경우 동일한 음원에 대한 허밍이라도 사용자에 따라 음의 높낮이, 템포 등이 다르고, 동일한 사용자라도 시간에 따라 허밍의 정확도에 상당한 편차가 존재할 수 있다. 따라서 다성음원과 사용자의 노래, 허밍 질의를 기반으로 하는 QbSH 시스템의 매칭엔진에 대한 설계는 이러한 입력 데이터의 부정확성을 미리 고려해야만 한다.

본 논문에서는 매칭엔진을 설계하는 단계에서 고려되어야 하는 부분과 추출되어진 특성 정보의 부정확성에 대한 극복 방안을 제시한다. 제안된 매칭엔진은 DTW 알고리즘 [10]을 기반으로 비대칭의 개념을 적용하였다. 추가적으로 가중치 계수(Weighting Coefficient), 크로마-스케일 표현기법(Chroma-Scale Representation), 보상기법(Compensation)을 결합하여 특성정보의 부정확성이 매칭엔진에 주는 영향을 최소화하였다. 즉, 피치 더블링(Pitch Doubling)과 피치 해빙(Pitch Halving) 오류의 영향을 줄이기 위한 피치 시퀀스 표현기법 및 피치 오류나 거리 오류에 대한 보상기법을 적용하였다. 제안한 매칭엔진은 구현 및 평가를 통해 그 성능수준을 검증하였다.

2장에서는 전체적인 QbSH 시스템의 구조에 대해 기술하였고, 3장에서는 다성음원 및 사용자의 질의에서의 특성정보 추출기법에 대해 기술하였다. 4장에서는 제안된 매칭엔진의 세부 알고리즘에 대해 기술하였고, 5장에서는 실험을 통해 제안된 QbSH 시스템의 세부 알고리즘 및 전체 시스템의 성능을 검증하였으며, 마지막으로 6장에서 결론을 맺었다.

2. QbSH 시스템 구조

다성음원 기반 QbSH 시스템은 사용자의 질의 데이터와 다성음원 데이터베이스의 음원 데이터, 두 가지 입력 데이터를 갖는다. QbSH 시스템은 두 입력에 대해 피치 시퀀스를 추출하여 특성정보로 사용하며, 매칭엔진에서 두 특성정보를 비교하여 가장 유사한 정보를 찾는 것이 QbSH 시스템의 기본 정의이다.

기존 QbSH 시스템에서 피치 시퀀스와 함께 노트 시퀀스, 피치 주기를 특성정보로 사용하는 경우도 있다 [11,12]. 그러나 기존 시스템은 대부분 단성음원을 입력 데이터로 사용하는 시스템이다. 실제 다성음원에서 신뢰성 있는 피치 노트 및 피치 주기를 추출하는 것은 매우 어렵다. 또한, 이를 통한 성능개선의 수준이 시스템의 부하 증가에 비해 주목할 만한 수준이 아니어서, 본 논문에서는 피치 시퀀스만을 활용하여 QbSH 시스템을 구성하였다.

그림 1은 다성음원 기반 QbSH 시스템의 구성도이다. 그림 1의 아래 부분은 다성음원에서 특성정보를 추출, 데이터베이스를 구축하는 부분이다. 그림 1의 위쪽 부분은 사용자의 질의에 대한 특성정보를 추출 부분과 매칭엔진 부분을 포함한다. 다성음원 데이터베이스 구축을 위해 MP3와 같은 다성음원에서 추출된 특성정보를 기반으로 “Feature DB”를 구성한다. QbSH 시스템의 매칭엔진은 사용자 입력 데이터의 피치 시퀀스와 가장 유사한 데이터를 “Feature DB”에서 찾아서 해당 곡 정보를 사용자에게 제공하게 된다.

3. 특성 정보 추출

일반적으로 질의 입력인 허밍이나 노래는 단성음원이며, 단성음원에서 음의 높낮이 정보인 피치 시퀀스 추출은 쉽게 접근이 가능하다. 실제 G.729 등의 음성 부호화기를 사용할 경우, 용이하게 구현할 수 있다.

다성음원에서의 멜로디 추출(Melody Transcription or Melody Extraction)은 단성음원의 피치 시퀀스 추출보다 상대적으로 어렵고, 정확도가 낮으며, 상대적으로 복잡한 처리 프로세서를 필요로 한다. 다성음원에서 멜로디 추출은 기본적으로 멀티 피치 추출(Multi-Pitch Extraction), 멜로디 인식 (Melody Identification) 및 후처리(Post-processing) 단계를 거쳐 구현된다. 한편, QbSH 시스템의 매칭엔진이 효율적으로 동작하기 위해서는 입력 데이터에서 추출한 특성정보의 형식이 동일해야 하고, 추출 주파수 및 피치 정보의 차원이 동일해야 한다.

제안한 QbSH 시스템은 MP3 파일을 사용하여 다성음원 데이터베이스를 구축하였다. 전처리 단계에서는 특성정보 추출을 위해 오디오 코덱에 의해 압축되어 있는 MP3 파일을 먼저 디코딩한다. 디코딩된 신호에 대해 8kHz로 다운 샘플링(Down-Sampling)을 수행한다. 이 데이터를 50% 중첩을 가지는 64ms 단위의 프레임으로 나누어, 특성정보 처리를 위한 기본 단위 형태로 구분하게 된다.

하나의 64ms 프레임에 대한 피치 주파수가 구해지면, 이를 식 (1)과 같은 세미톤(Semitone) 형식으로 표시할 수 있다. 일반적으로 세미톤은 정수로 표현하지만, 정확한 데이터의 표현 및 비교를 위해 실수 연산이 가능하도록 변환하였다.

$$Semitone_{real} = 12 \times \log_2\left(\frac{f}{440}\right) + 69 \qquad (1)$$

여기서 f는 피치의 주파수를 의미한다. 질의 입력과 다성음원에서 추출된 특성정보를 세미톤 형식의

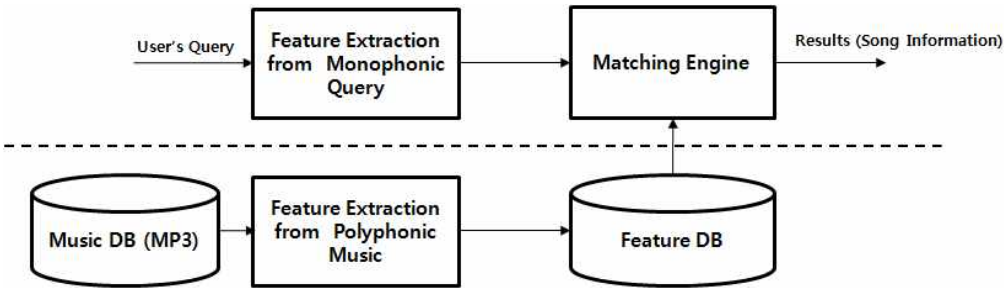

그림 1. 다성음원 기반 QbSH 시스템 구성도

표현한 데이터가 그림 1 매칭엔진의 입력이 된다.

3.1 사용자 질의에서의 피치 시퀀스 추출

사용자의 허밍 및 노래 질의에 대한 피치 시퀀스 추출을 위해 그림 2와 같은 시스템을 구성하였다. 먼저 사용자의 허밍 질의는 묵음구간과 같이 정보가 없는 구간이 존재한다. 실제 입력 데이터에 정보가 있는지 여부를 판단하기 위해 VAD (Voice-Activity Detection) 블록을 적용하였다. 이는 입력이 존재하는 구간의 피치 시퀀스 추출 오류보다, 입력이 없는 구간을 있는 것으로 판단하고, 피치 시퀀스를 추출하는 것이 성능에 보다 나쁜 영향을 미치기 때문이다.

VAD 블록에는 허밍과 노이즈 신호의 스펙트럼 비율(Spectrum Ratio)을 기반으로 하는 IMCRA (Improved Minima Controlled Recursive Averaging) 알고리즘 [13]을 사용하였다. 데이터의 정보가 없는 구간의 피치 값은 0으로 설정하였다.

VAD 이후에는 시간-주파수 영역에서의 자기상관(Time-Frequency domain AutoCorrelation, TFAC)을 이용한 피치추출 알고리즘을 적용하였다[14]. 이 알고리즘은 기존의 시간영역에서의 자기상관과 주파수 영역에서의 자기상관이 선형적으로 결합된 형태를 가진다. 일반적으로 시간영역과 주파수영역에서 피치 추출의 단점은 각각 피치 더블링과 피치 해빙 오류가 발생하는 것이다. 이러한 오류를 줄이기 위해 시간과 주파수 영역에서 자기상관을 이용하는 피치 추출 방법을 적용하였다.

3.2 다성음원에서의 피치 시퀀스 추출

그림 3은 다성음원에서 피치 시퀀스 추출을 위한 시스템 구성도이다. 이 시스템은 다중 피치 추출 모듈(Multi-Pitch Extraction)과 피치 트래킹(Pitch Tracking) 모듈로 구성된다. 다중 피치 추출 모듈은 먼저 신호에 포함되어 있는 다수의 피치 정보를 추출한다. 추출된 피치들의 하모닉 구조의 유무와 정확도에 따라 유효한 피치를 선택하고, 각각의 우선순위를 결정한다[15]. 피치 트래킹 모듈은 각 프레임간 기본 주파수의 연속성과 기본 주파수의 순위를 고려하여 피치 트래킹(Pitch Tracking)을 실행한다. 다성음원에서의 피치 시퀀스 추출은 다음의 과정으로 정의된다.

Step 1 : 다성음원 신호는 다수의 음원들이 혼합되어 있는 신호이며, 각 음원의 기본 주파수에 해당하는 하모닉 피크들의 조합이 존재한다. 기본 주파수를 추출하기 위해 스펙트럼에 나타나는 주파수 피크를 검출한다. 기본 주파수 후보를 정하기 위한 피크 사이의 간격은 식 (2)와 같이 정의된다.

$$\triangle[i,j] = LP[i] - LP[j] \tag{2}$$

검출된 주파수 피크의 위치를 $LP[\bullet]$ 로 정의한다.

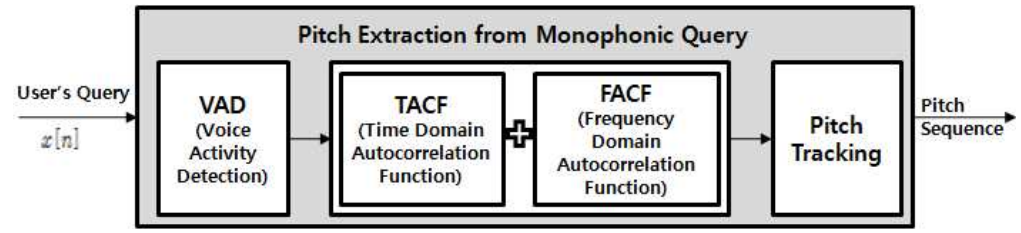

그림 2. 사용자 질의에서 피치 시퀀스 추출을 위한 시스템 구성도

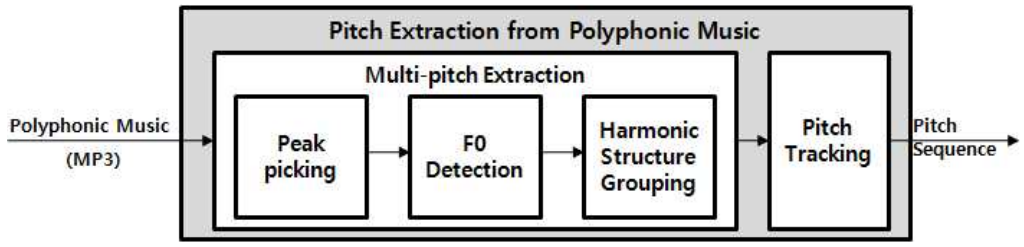

그림 3. 다성음원에서 피치 시퀀스 추출을 위한 시스템 구성도

$i = j+1,...,C,\ j = 1,...,C$ 이며, C는 해당 프레임에서 추출된 주파수 피크의 전체 개수이다.

Step 2 : 모든 기본 주파수가 이상적인 하모닉 구조를 가진다는 가정한다. 이상적 하모닉 피크 위치 h_p에 대해 실제 주파수 피크 $LP[\bullet]$가 존재하는지 확인한다.

Step 3 : 모든 $\triangle[i,j]$ 각각에 대해 이상적 하모닉 피크 위치 h_p에 대한 조건 만족 여부를 확인한다. 50% 이상의 하모닉 피크에서 조건이 만족되면, 해당 $\triangle[i,j]$를 기본 주파수 후보 $\widehat{F0}$로 결정한다.

Step 4 : 각 프레임 단위 기본 주파수 후보의 중요도를 결정하기 위해 AHS (Average Harmonic Structure) 값을 계산한다.

Step 5 : 각 프레임별 기본 주파수 후보의 피치 트래킹을 통해 주요 기본 주파수를 선택한다. 피치 트래킹 블록은 주변 프레임간의 기본 주파수의 연속성을 기반으로 한다.

4. 매칭엔진

매칭엔진은 다성음원 데이터에서 추출된 특성정보 $S_{DB}^{(i)}$ 정보로 구성된 Feature DB상의 정보인 $INFO^{(i)}$와 사용자의 질의에서 추출된 S_q사이에서 가장 유사한 정보를 찾아서 그 결과를 제공하는 부분이다. 이를 위해 매칭엔진은 두 데이터 S_q와 $S_{DB}^{(i)}$ 간의 불일치에 대해서 강인한 특성을 지녀야 한다. 표 1은 논문의 매칭엔진의 기술과 관련되어 사용되는 기호에 대한 정의이다.

4.1 입력 데이터에 대한 피치 추출의 부정확성

입력되는 두 개의 데이터를 비교하는 매칭엔진에서는 일반적으로 입력 데이터의 크기, 형태, 표현 방법 등의 불일치로 인해 발생하는 오류에 대해 설계단계에서 이를 고려해야 한다. QbSH 시스템의 매칭엔진에 입력되는 피치 시퀀스는 악보에 있는 멜로디를 기반으로 생성되며, 두 입력에 대한 각각의 피치 시퀀스를 추출하는 단계에서부터 상당부분 왜곡이 발생한다. 다성음원에서 피치 정보를 추출하는 경우,

표 1. 매칭엔진의 정의 및 설명을 위해 사용되는 기호 리스트

기 호	정 의
S_q	pitch sequence of query
$S_{DB}^{(i)}$	pitch sequence of ith song stored in DB
$INFO^{(i)}$	information of the pitch sequence
I	the number of songs in DB
$L(\bullet)$	length of pitch sequence
$d_{DTW}(P,Q)$	distance between two sequences P and Q based on DTW
$d_M(P,Q)$	distance between two sequence P and Q computed in matching engine
$d(\bullet)$	arbitrary distance metric
α	weight coefficient for DTW
c	compensation coefficient
$\Psi(\bullet)$	pre-processing of matching engine

이러한 왜곡현상에 대한 극복이 특히 중요하다.

다성음원은 메인 멜로디를 포함하여 다양한 악기의 소리를 포함하고 있으며, 이렇게 다양한 소리가 조합된 음원에서 각 음원을 개별적인 오디오 신호로 완벽하게 분리해 내는 것은 거의 불가능하다. 이는 메인 멜로디의 소리와 악기 등의 주변 소리가 동시에 발생하고, 들리기 때문이다. 또한 메인 멜로디의 피치 시퀀스를 완벽하게 추출한다 하더라도, 메인 멜로디가 포함되어 있지 않은 프레임 구간에서는 함께 존재하는 다른 소리가 메인 멜로디의 피치 시퀀스로 대신 추출되는 현상이 발생한다. 이러한 부분이 다성음원에서 메인 멜로디의 피치 시퀀스의 추출을 어렵게 만드는 요인이다. 따라서 다성음원에 대한 피치 시퀀스의 추출은 메인 멜로디와 주변 멜로디의 혼재상황을 충분히 고려해야 한다.

3.1절에 언급된 피치 더블링과 피치 해빙 오류 역시 부정확한 피치 시퀀스를 만드는 요인이다. 일반적으로 피치 더블링과 해빙 효과에 의한 오류는 피치 추출이 주기적 신호에 대해 수행되기 때문에 발생하며, 완벽하게 회피할 수는 없다.

한편, 사용자의 노래, 허밍 질의 입력에 대한 피치 시퀀스 추출의 경우, 기술적 접근의 어려움보다는, 실제 악보에 있는 원래의 멜로디 정보와 전혀 다른 정보의 데이터가 쉽게 유입된다는 점을 주목해야 한다. 일반적으로 사용자는 악보에 있는 정보대로 전혀 노래나 허밍을 하지 않는 경우가 많다. 대부분의 사

용자는 잘못된 템포로, 부정확하고, 상대적인 피치 값을 가지고 노래 또는 허밍을 한다. 때로는 각 음의 길이가 완전히 틀리기도 하며, 노래의 한 소절중 상당 부분을 빠트려 전혀 다른 노래를 입력하기도 한다.

QbSH 시스템에는 각 입력 데이터에 대한 특성추출 단계를 통해 피치 시퀀스가 추출되기 전 이미 입력 데이터에 상당한 오류가 존재하며, 추가적으로 시스템의 각 단계에 적용된 동작 알고리즘에서도 일정 부분 오류가 존재한다.

4.2 매칭엔진에서의 전처리 과정

입력 데이터가 가지는 데이터의 오류, 즉 부정확성에 대해 강인한 특성을 확보하기 위해 매칭엔진에서는 입력 데이터에 대해 전처리 과정이 요구된다. 본 절에서는 보상계수의 적용 및 크로마-스케일 표현기법에 의한 전처리 과정을 제안한다.

QbSH 시스템에서 매칭엔진 목적은 질의 입력의 특성정보와 데이터베이스에 저장된 특성정보를 비교하여 사용자가 허밍하거나 노래한 것으로 추정되는 음원을 찾는 것이다. 이를 수식으로 표현하면 식 (3)과 같다.

$$\hat{i} = \arg\min d_M(S_q, S_{DB}^{(i)}) \tag{3}$$

여기서 특성정보 데이터베이스에 피치 시퀀스 $S_{DB}^{(i)}$가 이미 저장되어 있는 것으로 가정한다. 매칭엔진 $d_M(\bullet)$에 의해 구해지는 입력 피치 시퀀스와 데이터베이스에 있는 피치 시퀀스간의 차, 즉 두 데이터간 거리(Distance)는 식 (4)와 같다.

$$d_M(S_q, S_{DB}^{(i)}) = \min d_{DTW}(\Psi(S_q + c), \Psi(S_{DB}^{(i)})) \tag{4}$$

여기서 c는 보상계수(Compensation Coefficient)이며, $\Psi(\bullet)$는 전처리를 위한 함수이다. $\hat{i}$가 결정되면, $\hat{i}$ 번째 피치 시퀀스 정보 $INFO^{(i)}$를 구할 수 있다. $S_{DB}^{(i)}$ 값은 피치 더블링과 피치 해빙 효과로 인해 부정확 해질 수 있으며, 이러한 효과에 의해 세미톤으로 표시된 값은 원래의 피치 값에서 한 옥타브를 나타내는 수인 12만큼 커지거나 작아지게 된다. 이러한 부정적 효과에 의한 피치 값의 변화를 최소화하기 위하여, 피치 값을 12로 나누고 그 나머지 값을 특성정보로 사용하는 크로마-스케일 표현기법을 적용한다. 따라서 $\Psi(\bullet)$는 입력 시퀀스의 나머지 값의 함수가 된다.

사용자 질의 입력에 의한 피치 시퀀스 역시 어느 정도 수준의 보상이 필요하며, 이는 사용자가 악보에 있는 원래의 피치 값과 동일하게 노래하거나 허밍을 수행하여 절대적으로 올바른 피치 값을 입력할 수는 없기 때문이다. 제안된 시스템은 식 (4)와 같이 보상계수 c를 피치 시퀀스에 더하여 입력 데이터에 대한 보상을 수행한다. 전처리 단계에서의 피치 값의 범위를 고려하여, c의 범위는 $0 \le c \le 12$ 로 제한하며, 정수 값의 c를 사용하였다.

4.3 비대칭적 DTW 알고리즘

QbSH 시스템은 일반적으로 사용자의 허밍 질의에 대한 시간적 변화가 큰 동시에, 부정확한 질의가 입력됨으로 해서 검색의 정확도가 감소할 수 있다. 이러한 입력 데이터의 특성을 반영하여 시간적 변화나 부정확한 템포에 강인한 특성을 제공하는 DTW 알고리즘을 매칭엔진의 기본적인 동작 개념으로 적용하였다.

일반적으로 DTW 알고리즘은 음성인식, 화자인식 등에 널리 적용되는 알고리즘으로, 서로 길이가 다른 두 패턴간의 유사도를 측정하기 위한 대표적인 기법이다. 이를 위해 삽입(Insertion) 및 제거(Deletion) 등의 방법으로 경로별 비교 패턴의 유사도를 측정한다. DTW 알고리즘을 적용하기 위한 기본 전제는 다음과 같다.

(1) 시작점-끝점 제약 조건 : 입력 데이터 및 비교 대상 데이터의 시작점과 끝점을 일치 시키고 비교를 진행

(2) 단조 증가 제약 조건 : 최적 경로는 항상 단조 증가

(3) 국부 연속 제약 조건 : 한 비교점에 도달하기 위한 경로의 제한을 두어 시간상에서 지나치게 경로가 축소되거나 팽창하는 것을 제한

(4) 전역 경로 제약 조건 : 서로 다른 길이를 갖는 입력 및 비교 데이터간 전 구간에 걸친 허용 가능한 영역을 제한

(5) 기울기 가중치 조건 : 국부 경로의 거리 계산시, 동일한 가중치를 주지 않고 기울기에 따라서 서로 다른 가중치를 주어, 시간에 비해 비합리적으로 변하는 것을 방지

그러나, DTW 알고리즘이 QbSH 시스템에 적용될 경우, 실제 질의 데이터의 피치 시퀀스를 다성음원의 특성정보 데이터베이스에 있는 모든 피치 시퀀스와 시간축상의 변화를 기반으로 계속적으로 비교하기 때문에 연산량이 과도하여 많은 검색 시간이 요구되고, 동시에 우수한 검색 정확도를 담보하기 힘들다는 문제점이 발생한다.

또, QbSH 시스템에서는 사용자의 허밍 질의 데이터의 길이가 비교대상인 다성음원에서 추출된 특성정보 데이터에 비해 상당히 짧다는 부분도 설계 및 구현 과정에서 고려되어야 한다. 즉, 일반적으로 사용자가 음원 전체를 허밍하거나 노래할 수 없기 때문에, 사용자의 질의 입력에 의한 피치 시퀀스는 “Feature DB”에 있는 다성음원에서 추출된 피치 시퀀스 길이에 비해 짧다. 실제 사용자가 허밍 할 수 있는 일반적인 시간은 10초 이내이며, 매칭엔진은 입력 질의가 어떤 곡의 일부라는 가정하에 동작하게 된다.

이러한 가정과 시스템의 환경을 고려하여 기존 DTW 알고리즘의 전제 조건에서 “(1) 시작점-끝점 제약 조건”을 제안된 QbSH 시스템에서는 적용하지 않는다. 이는 DTW 경로(Path)가 임의의 시작점에서 시작하여 임의의 점에서 끝날 수 있다는 것을 의미한다. 이러한 접근을 통해 검색을 위한 연산량 축소에 따른 검색 시간의 감소시킬 수 있으며, 동시에 검색 성능의 개선이 가능하다.

비대칭적 DTW 알고리즘에 대한 정의를 위해 먼저 임의의 벡터 P를 $P=[P(1)P(2)...P(K)]$ 로 표시하고, $P=\Psi(S_q)+c$, $Q=\Psi(S_{DB}^{(i)})$, $L(S_q)=K$, $L(S_{DB}^{(i)})=L$ 로 가정한다. 또, 거리 함수 (Distance Metric) $D_1=[D_1(k,l)]$ 와 $D_2=[D_2(k,l)]$ 를 $k=1,2,...,K$ 및 $l=1,2,...,L$에 대해 정의한다. 이 경우 두 입력 시퀀스 P와 Q에 대한 $d_{DTW}(P,Q)$값은 다음과 같은 단계로 구해진다.

Step 1 : 두 입력 시퀀스 P와 Q 그리고 S_q에 대해, $k=1,2,...,K$ 그리고 $l=1,2,...,L$인 경우, $D_1(k,l)$는 식 (5)와 같이 정의된다.

$$D_1(k,l)=\begin{cases}0 & \text{if } S_q(l)=0\\ d(P(k),Q(l)) & otherwise\end{cases} \tag{5}$$

Step 2 : $l=1,...,L$의 초기조건에서, $D_2(k,l)$은 식 (6)과 같이 정의된다.

$$D_2(1,l)=D_1(1,l) \tag{6}$$

Step 3 : $k=2,...,K$의 조건에서 $D_2(k,l)$은 식 (7)과 같이 정의된다.

$$D_2(k,1)=D_2(k-1,1)+D_1(k+1) \tag{7}$$

Step 4 : $k=2,...,K$ 그리고 $l=2,...,L$의 조건에서

· $k=2$ 혹은 $l=2$인 경우, $D_2(k,l)$은 식 (8)과 같이 정의된다.

$$D_2(k,l)=D_1(k,1)+\min(D_2(k,l-1),\\ D_2(k-1,l),D_2(k-1,l-1)) \tag{8}$$

· $k=2$ 혹은 $l=2$ 외의 경우, $D_2(k,l)$은 식 (9)와 같이 정의된다.

$$\begin{aligned}D_2(k,l)=\min(&D_1(k,l)+D_2(k-1,l-1),\\ &D_1(k,l)+D_2(k-1,l-2),\\ &\alpha\times D_1(k,l)+D_2(k-2,l-1))\end{aligned} \tag{9}$$

Step 5 : 최종 거리함수 $d_{DTW}(P,Q)$는 식 (10)과 같이 구해진다.

$$d_{DTW}(P,Q)=\min_{(l=[K/2],...,L)} D_2(K,l) \tag{10}$$

D_1은 각 프레임별 거리를 의미하며, 이때 $D_2(1,l)=D_1(1,l)$ 로 초기 값을 정의할 수 있다. 최종적으로는 전체 거리를 의미하는 $D_2(k,l)$ 을 구하고, 그 중에서 최소 왜곡을 가지는 경로를 구할 수 있다. 이때 $l=[K/2]$ 인 경우가 왜곡이 발생하는 가장 빠른 경로를 의미하며, $l=[K/2]$ 보다 작은 경우의 경로는 실제 의미가 없는 경로이므로 무시한다.

DTW $d_{DTW}(P,Q)$경로의 결정은 상관계수 α를 가지고 이루어지며, $\alpha=1$ 인 경우는 DTW 알고리즘은 대칭적 (Symmetric)이 되어 모든 경로는 동일한 비중을 가진다. 반면 $\alpha\neq 1$ 의 경우는, DTW 알고리즘이 비대칭적이 (Asymmetric) 된다[16].

본 논문에서는 다성음원 기반의 QbSH 시스템의 매칭엔진에 비대칭적 DTW 알고리즘을 적용하였다. 이는 기존 DTW 알고리즘과는 달리 DTW 경로가 어느 프레임에서든 시작할 수 있으며, 최소의 왜곡을 갖는 경로를 따라 어느 프레임에서나 그 경로가 끝날 수 있다는 것을 의미한다. 이를 통해 매칭엔진은 하

나의 음원에 대한 데이터 비교를 처음부터 끝까지 수행하는 것이 아니라, 최소 왜곡을 가지는 임의의 경로를 따라 그 연산을 수행하여, 최소 왜곡을 구하게 된다.

비대칭적 DTW가 적용된 QbSH 시스템에서는 피치가 추출되지 않는 프레임에 대해서는 그 거리를 계산하지 않는다. $S_{DB}^{(i)}$에 있는 모든 엘리먼트(Element)는 피치 값을 가지며, 0의 엘리먼트는 없다. 그러나 S_q에서는 0의 엘리먼트가 존재할 수 있으며, 이 0의 값에 대해서는 거리를 계산하지 않고 0으로 설정한다.

이러한 비대칭적 DTW 알고리즘을 QbSH 시스템에 적용할 경우, 연산량도 감소하는 동시에 검색 정확도 및 검색시간 측면에서 우수한 성능의 확보가 가능하다. 실제 동일한 질의 데이터 및 음원 DB 기반의 실험에서 비대칭적 DTW 알고리즘은 DTW 알고리즘 대비 연산량이 1/50로 감소하는 것을 확인하였다. 동시에 Roger Jang's correction을 DB로 사용하고, 정답이 검색 결과 상위 10위내에 포함되는 것을 기준으로 검색 정확도를 측정한 경우, 비대칭적 DTW는 95.0%로 DTW 84.2% 정확도에 대해 우수한 것을 확인할 수 있었다.

4.4 거리 함수(Distance Metric)

최종적인 매칭엔진의 검색 정확도는 거리 함수 $d(\bullet)$에 의해 결정되며, 이 값을 계산하기 위해 일반적으로 절대 값 혹은 그 제곱의 값이 DTW 알고리즘 기반의 QbSH 시스템에 적용되어 왔다. 이러한 거리 함수는 전형적인 DTW 알고리즘의 프레임워크를 적용하는 QbSH 시스템뿐만 아니라, 다양한 형태의 변형된 DWT 알고리즘을 적용한 시스템에도 유사하게 적용되었다. 검색의 정확도 향상을 위한 Multiple-Phase DTW, 기존 DTW의 연산의 복잡도를 줄이기 위한 Multiple Level Data Abstraction 알고리즘 및 Voiceless Region Bypassing 알고리즘 등을 갖는 시스템에도 기존의 거리함수가 대부분 적용되었다[16-18].

그러나 특성정보 데이터베이스에 있는 피치 시퀀스는 일반적으로 왜곡된 정보를 포함하고 있기 때문에, 보다 우수한 성능을 확보하기 위해서는 이러한 왜곡에 둔감한 거리 함수가 요구된다.

본 논문에서는 다양한 거리 함수를 제안하여 다성음원 기반의 QbSH 시스템에서 최대 성능을 가지는 거리 함수를 정의하였다. 식 (11)과 (12)는 일반적으로 사용되는 거리 함수의 정의이며, 실험을 통한 성능의 비교를 위해 사용되었다.

$$d_{|\bullet|}(a,b) = |a-b| \tag{11}$$

$$d_{|\bullet|^2}(a,b) = |a-b|^2 \tag{12}$$

매칭엔진의 우수한 성능확보를 위해 두 인수 차의 slope가 그 값의 차이에 따라 감소하여 왜곡에 의한 영향을 적게 받을 수 있는 거리 함수가 요구된다. 이를 위해 식 (13)~(15)를 제안하였다. 식 (13)의 거리 함수는 그 최대값을 γ로 제한하여 그 절대적인 차의 값이 γ를 넘으면, 거리 값이 γ로 고정된다. 식 (14)는 logistic loss 기반으로 거리 함수를 정의하고 있으며[19], 식 (15)는 sigmoid 함수를 적용하고 있다. 제안한 거리 함수 식 (13)~(15)를 QbSH 시스템에 적용하여 그 성능을 평가하고, 최적의 거리 함수를 설정하였다.

$$d_{HINGE}^{(\lambda)}(a,b) = \begin{cases} |a-b| & \text{if } |a-b| < \gamma \\ \gamma & otherwise \end{cases} \tag{13}$$

$$d_{LOG}(a,b) = \log(1+|a-b|) \tag{14}$$

$$d_{SIG}(a,b) = \frac{1}{1+\exp(-|a-b|)} - 0.5 \tag{15}$$

5. 구현 및 성능 평가

5.1 기존 QbSH 시스템과의 성능 비교

5.1.1 기존 단성음원 기반 QbSH 시스템과의 성능 비교

제안한 QbSH 시스템의 전체 성능을 비교 검증하기 위해서는 우선 기존 QbSH 시스템과 동일한 평가 환경을 구축하여야 한다. 평가 환경의 기본적인 요소는 동일한 입력인 질의 데이터와 검색 대상인 음원 데이터의 구축이다. 이를 위해 MIREX (Music Information Retrieval Evolution Exchange)[20] QbSH Task의 평가 환경을 적용하였다. 제안한 시스템과 MIREX 2010 QbSH Task에 출품된 QbSH 시스템 및 DTW 알고리즘 기반 QbSH 시스템의 성능을 비교하여 그 성능을 평가하였다.

제안한 다성음원 기반 QbSH 시스템을 기존 단성음원 기반 QbSH 시스템과 비교 평가하기 위해 먼저 제안한 시스템의 "다성음원 피치 시퀀스 추출 (Pitch

Extraction From Polyphonic Music)" 부분을 제거하고, MIREX QbSH Task에서 공개한 Roger Jang's Collection을 단성음원 데이터베이스로 적용하였다. 이 데이터베이스는 48개의 음원 데이터와 4,431개의 허밍 질의 데이터를 포함하고 있다. 추가로 2,000개의 MIDI 노이즈 데이터를 추가하여 실험 환경을 구축하였다.

QbSH 시스템의 검색 정확도를 측정하기 위해 질의 데이터와 동일한 음원이 결과로 제시된 상위 10위 안에 포함된 경우, 즉 정답이 유사도가 높은 상위 10개 안에 존재할 경우, 정답으로 인정하는 Top 10 Hit Rate를 사용하였다.

표 2는 Roger Jang's Collection을 기반으로 비대칭적 DTW 알고리즘을 적용한 제안 시스템과 기존 QbSH 시스템과의 검색 정확도를 비교하여 보여준다. 성능 비교를 통해 비대칭적 DTW 알고리즘 기반의 QbSH 시스템이 단성음원 데이터베이스 환경에서도 우수한 성능을 보임을 확인할 수 있다. 또, 기존 DTW 알고리즘이 적용된 QbSH 시스템에 비해서도 상당히 우수한 성능을 가지는 것을 알 수 있다.

5.1.2 비대칭적 DTW 알고리즘의 연산량 비교

제안한 비대칭적 DWT 알고리즘의 연산량을 정량화하고, 이를 기존 DTW 알고리즘 연산량과 비교하여 연산량의 감소를 확인하였다. 제안한 비대칭적 DTW 알고리즘과 기존 DTW 알고리즘의 연산량 정량화는 다음의 과정을 통해 확인할 수 있다.

표 2. 비대칭적 DTW 알고리즘과 기존 알고리즘간 성능비교 (Top 10 Hit Rate)

QbSH 시스템(MIREX Sub-code)	성 능
HAFR1(2010)	0.771
JY1(2010)	0.967
JY2(2010)	0.956
YF1(2010)	0.947
YF(2010)	0.912
HAFR(2010)	0.77
CSJ1(2010)	0.94
CSJ2(2010)	0.90
DWT 기반 QbSH 시스템	0.842
Proposed QbSH 시스템 (Asymmetric-DTW)	0.95

Step 1 : 비대칭적 DTW 알고리즘에서 질의 입력의 특징 데이터의 길이를 L, 음원데이터의 특징 데이터의 길이를 M으로 정의한다.

Step 2 : 비대칭적 DTW 알고리즘의 연산량은 L×M 크기의 매트릭스의 각 원소 값을 구하는 것과 같다. 따라서 원소 값을 구하기 위해 DTW 경로의 거리를 계산하고, 세 개의 경로중 최소값을 찾는 연산량을 C로 정의한다.

Step 3 : 비대칭적 DTW 알고리즘의 연산량 N_{A-DTW}는 식 (16)과 같이 정의된다.

$$N_{A-DTW} = L \times M \times C \tag{16}$$

Step 4 : 기존 DTW 알고리즘의 연산량은 음원 데이터 M의 길이를, K의 길이를 갖는 데이터로 나누어서 반복적으로 DTW 연산을 수행하는 것으로 정의한다. K의 길이가 M의 1/20의 크기로 가정한다. 또 DTW 연산을 K/10 마다 반복적으로 수행된다고 가정한다. 이 경우 1회 수행한 연산량 N은 식 (17)과 같다.

$$N = K \times L \times 200 \times C = 10\,LMC \tag{17}$$

Step 5 : (식) 17에 대해 K의 길이를 M번 달리하면서 반복한다고 가정한다. 이 경우, DTW 알고리즘의 최종 연산량 N_{DTW}는 식 (18)과 같이 정의된다.

$$N_{DTW} = K \times L \times 200 \times C \times 5 = 50\,LMC \tag{18}$$

일반적인 질의 데이터와 다성음원 데이터의 크기를 고려하고, 반복적으로 수행되는 DTW 알고리즘의 정의를 반영하면, 비대칭적 DTW 알고리즘의 연산량이 기존 DTW 알고리즘의 약 1/50에 해당하는 것을 확인할 수 있다.

5.1.3 다성음원의 피치 시퀀스 추출 성능 검증

단성음원 기반 QbSH 시스템과의 성능 비교를 위해 그 기능이 제한된 다성음원에서의 피치 시퀀스 추출 모듈, 즉 메인 멜로디 추출 모듈은 MIREX의 Melody Extraction Task와 정확히 그 정의가 일치한다. 따라서 제안한 QbSH 시스템에 포함되는 다성음원에서의 피치 시퀀스 추출 모듈의 성능은 표 3을 통해 확인 할 수 있다.

표 3. 다성음원에서의 피치 시퀀스 추출의 정확도

알고리즘 (MIREX Sub-code)	Accuracy			Voicing	
	Overall	Raw Pitch	Raw Chroma	Recall	False
HJ1(2010)	0.6133	0.7683	0.7976	0.7343	0.2099
TOOS1(2010)	0.5368	0.6285	0.7338	0.7958	0.3277
JJY2(2010)	0.7194	0.7964	0.8527	0.9365	0.5011
SG1(2010)	0.6993	0.7522	0.7822	0.8055	0.2318
CL1(2009)	0.7659	0.8507	0.8635	0.9563	0.6928
DR1(2009)	0.7569	0.8140	0.8344	0.9034	0.4120
HJC1(2009)	0.4740	0.6386	0.7361	0.4573	0.7126
JJY(2009)	0.7511	0.8326	0.8699	0.8676	0.3585
KD(2009)	0.8630	0.8711	0.8762	0.9163	0.1578
MW(2009)	0.7082	0.8225	0.8640	0.9994	0.9383
PC(2009)	0.8250	0.8286	0.8340	0.8464	0.1550
RR(2009)	0.7074	0.7688	0.8509	0.9200	0.4795
TOOS(2009)	0.5249	0.6103	0.7178	0.9991	0.9318
Proposed	0.857	0.8692	0.8803	0.9351	0.2388

다성음원의 피치 시퀀스 추출 성능의 검증을 위해 MIREX Melody Extraction Task에서 성능 평가를 위해 적용하는 ADC2004 데이터베이스를 사용하였다. ADC2004는 음원 특성 추출 연구에 널리 사용되는 데이터베이스로 총 20개의 데이터로 구성되어 있다. 정답으로 사용되는 멜로디의 기본 주파수는 SMSTools를 사용하여 5.8ms 단위로 추출되어 제공된다.

표 3은 ADC2004를 기반으로 하는 피치 시퀀스 추출의 정확도를 나타낸다. 비교 대상은 MIREX 2009 및 MIREX 2010에 Melody Extraction Task에 출품한 알고리즘이다. 비교 결과를 통해 제안된 피치 시퀀스 추출 알고리즘의 성능이 상대적으로 우수함을 확인할 수 있다.

5.2 다성음원 기반 QbSH 시스템 구현 환경

다성음원 기반 QbSH 시스템의 구현 및 시스템 전체의 성능 테스트를 위해 450곡의 다성음원을 기반으로 데이터베이스를 구축하였다. 이 데이터베이스에 포함된 전체 오디오 데이터의 길이는 총 28시간이며, 개별 오디오 데이터의 길이는 34초부터 396초까지 다양하게 존재한다. 또 450곡의 데이터베이스는 발라드, 댄스, 팝, R&B, 록, 트롯, 동요, 캐럴 등의 다양한 장르의 음원이 포함되어 있다.

이러한 다성음원 데이터베이스를 기반으로 사용자의 10초 내외의 허밍 또는 노래 입력에 대해 입력과 동일한 곡의 제목을 검색하는 QbSH 시스템을 그림 4와 같이 구현하였다. 제안된 QbSH 시스템은 사용자의 질의와 가장 유사한 20개의 음원을 순위별로 제공하며, 제목, 가수 및 노래 가사 등의 정보도 사용자에게 함께 제공한다.

성능평가를 위하여 성별과 연령이 다양한 32명의 지원자를 통해 약 10~12초 길이의 1,000개의 질의 데이터를 취득하였다. 지원자는 14명의 여자와 18명의 남자로 구성되며, 데이터베이스에 존재하는 곡 중에서 한 곡을 선택하여, 곡의 임의의 부분에 대해 허밍 혹은 노래를 입력하여 질의 데이터를 생성하였다. 최종적으로 취득된 질의 데이터는 298개의 허밍 질의와 702개의 노래 질의로 구성되었으며, 각 지원자의 연령, 음악적 전문성 및 질의의 정확성에 대한 정보를 추가로 반영하여 최종 평가 환경을 구성하였다.

제안한 QbSH 시스템의 성능의 평가를 위해 MIREX에서 QbSH 시스템의 평가를 위해 사용하고 있는 역순위 평균(Mean Reciprocal Rank, MRR) 방법을 적용하였다. 이 평가 방법은 QbSH 시스템의 정확도, 즉 입력 질의에 대해 올바르게 검색된 곡의 정확도를 평가하는 방법으로 정보 검색 시스템의 성능 평가에 일반적으로 사용된다.

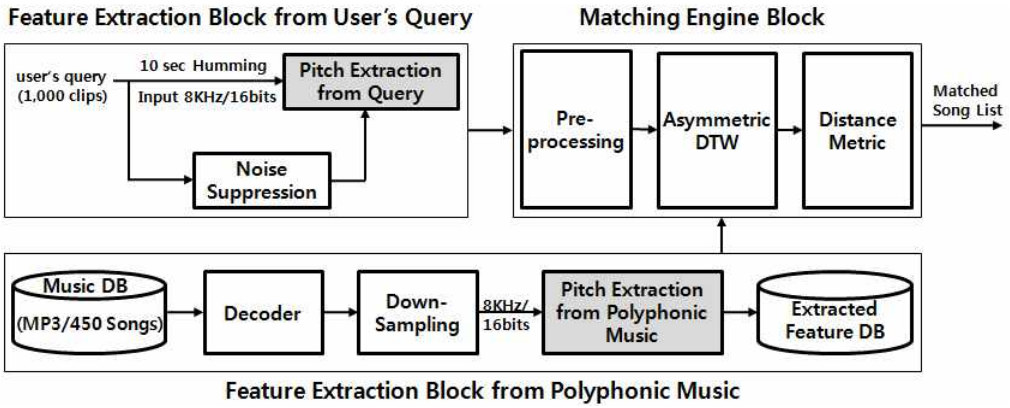

그림 4. 전체 QbSH 시스템 구성도

성능의 평가를 위해 사용자에게 허밍, 노래 질의와 가장 유사한 상위 20개의 음원을 제공하고, 그중에서 정답에 해당하는 곡의 순위에 대한 역순위 평균을 적용하여 MRR 값으로 검색의 정확도를 측정하였으며, 검색 결과가 1위, 상위 10위, 상위 20위에 포함되는 비율을 백분율로 함께 측정하였다.

5.3 다성음원 기반 QbSH 시스템 성능

비대칭적 DTW 및 다양한 거리함수를 적용한 다성음원 기반의 QbSH 시스템의 성능은 표 4와 같다. $\alpha=3$, $\lambda=2$, 거리 함수가 $d^{(2)}_{HINGE}(\cdot)$의 경우, MRR이 0.718로 가장 좋은 성능을 보였다. 비록 실험 대상인 데이터베이스가 달라 직접적인 성능의 비교는 힘들지만, MRR 0.718의 성능은 427개의 자체적인 다성음원 데이터베이스를 기반으로 한 기존 연구의 검색성능 MRR 0.578 연구보다 월등히 우수한 결과이다

표 4. 거리 함수와 α에 따른 QbSH 시스템 성능(MRR)

$d(\cdot)$ \ α	1	2	3	4
$d_{\lvert\cdot\rvert}(\cdot)$	0.537	0.627	0.631	0.620
$d_{\lvert\cdot\rvert^2}(\cdot)$	0.456	0.505	0.517	0.520
$d^{(1)}_{HINGE}(\cdot)$	0.456	0.666	0.647	0.625
$d^{(2)}_{HINGE}(\cdot)$	0.606	0.713	**0.718**	0.712
$d^{(3)}_{HINGE}(\cdot)$	0.601	0.696	0.708	0.698
$d^{(4)}_{HINGE}(\cdot)$	0.578	0.674	0.686	0.679
$d_{LOG}(\cdot)$	0.542	0.674	0.672	0.658
$d_{SIG}(\cdot)$	0.581	0.704	0.707	0.696

[21]. 또한 200개의 오디오 데이터베이스를 기반으로 하는 기존 시스템 [22]보다도 매우 높은 검색 성능을 보여주고 있다.

표 4의 결과에서 $\alpha=3$인 경우, 거의 대부분의 거리 함수에서 QbSH 시스템의 성능이 우수한 것으로 측정되었다. 한편, $\alpha=2$, $\alpha=4$인 경우는 $\alpha=3$의 경우에 비해 심각하게 낮은 성능을 보이지는 않지만, $\alpha=1$ 인 경우, 즉 DTW 알고리즘이 대칭인 경우는 분명한 성능의 저하를 보이고 있다. 이러한 성능 분석을 통해 $\alpha\neq1$과 같이 비대칭적 DTW 알고리즘이 적용된 QbSH 시스템의 성능이 우수하다는 것을 확인할 수 있다. 또, 제안한 3개의 거리 함수는 기존 거리 함수 $d_{\lvert\cdot\rvert}(\cdot)$, $d_{\lvert\cdot\rvert^2}(\cdot)$ 보다 우수한 성능을 보이는 것을 확인할 수 있다.

다양한 거리 함수에 따른 QbSH 성능 평가에서 입력된 질의에 대해 입력과 동일한 음원이 가장 유사한 음원, 즉 1위로 제시된 경우, 유사도가 높은 10개의 곡 안에 포함된 경우, 유사도가 높은 20위 안에 포함된 경우의 비율은 표 5와 같다. 표 5도 표 4와 동일하게 $\alpha=3$, $d^{(2)}_{HINGE}(\cdot)$의 경우 가장 높은 성능을 보인다. 거리 함수와 α에 따른 Top1, Top10, Top20의 포함 비율은 표 4의 전체 시스템의 MRR 성능과 거의 동일한 형태의 패턴을 보여준다.

표 6은 보상 계수에 따른 QbSH 시스템의 성능을 보여주고 있다. 기본적으로 c를 정수로 정의하여 사용하며, c값에 따른 보상의 많고, 적음에 따라 그 결과는 다르게 나타난다. 표 6에서 c는 Matlab 표현으로 정의되어 있으며, $\alpha=3$, $d^{(2)}_{HINGE}(\cdot)$의 경우에 대해 실험이 수행되었다. 표 6에서 볼 수 있듯이 보다

표 5. 거리 함수와 α에 따른 Top1/10/20 포함 비율 (%)

α	$d(\cdot)$	Top1	Top10	Top20
1	$d_{\|\cdot\|}(\cdot)$	0.479	0.657	0.719
	$d_{\|\cdot\|^2}(\cdot)$	0.398	0.583	0.638
	$d^{(1)}_{HINGE}(\cdot)$	0.388	0.586	0.656
	$d^{(2)}_{HINGE}(\cdot)$	0.546	0.723	0.773
	$d^{(3)}_{HINGE}(\cdot)$	0.543	0.723	0.770
	$d^{(4)}_{HINGE}(\cdot)$	0.516	0.702	0.752
	$d_{LOG}(\cdot)$	0.480	0.676	0.725
	$d_{SIG}(\cdot)$	0.517	0.707	0.759
2	$d_{\|\cdot\|}(\cdot)$	0.569	0.646	0.794
	$d_{\|\cdot\|^2}(\cdot)$	0.449	0.620	0.673
	$d^{(1)}_{HINGE}(\cdot)$	0.610	0.779	0.822
	$d^{(2)}_{HINGE}(\cdot)$	0.668	0.808	0.843
	$d^{(3)}_{HINGE}(\cdot)$	0.643	0.790	0.834
	$d^{(4)}_{HINGE}(\cdot)$	0.615	0.786	0.822
	$d_{LOG}(\cdot)$	0.618	0.780	0.814
	$d_{SIG}(\cdot)$	0.656	0.799	0.838
3	$d_{\|\cdot\|}(\cdot)$	0.572	0.759	0.794
	$d_{\|\cdot\|^2}(\cdot)$	0.465	0.642	0.691
	$d^{(1)}_{HINGE}(\cdot)$	0.583	0.759	0.813
	$d^{(2)}_{HINGE}(\cdot)$	0.671	0.814	0.842
	$d^{(3)}_{HINGE}(\cdot)$	0.659	0.803	0.834
	$d^{(4)}_{HINGE}(\cdot)$	0.632	0.795	0.822
	$d_{LOG}(\cdot)$	0.613	0.780	0.814
	$d_{SIG}(\cdot)$	0.658	0.804	0.834
4	$d_{\|\cdot\|}(\cdot)$	0.557	0.643	0.788
	$d_{\|\cdot\|^2}(\cdot)$	0.466	0.751	0.694
	$d^{(1)}_{HINGE}(\cdot)$	0.557	0.811	0.798
	$d^{(2)}_{HINGE}(\cdot)$	0.663	0.809	0.839
	$d^{(3)}_{HINGE}(\cdot)$	0.641	0.789	0.836
	$d^{(4)}_{HINGE}(\cdot)$	0.623	0.778	0.821
	$d_{LOG}(\cdot)$	0.598	0.778	0.812
	$d_{SIG}(\cdot)$	0.642	0.804	0.837

표 6. 보상 계수 (Compensation Coefficient)에 따른 QbSH 시스템 성능

c	MRR	Top1	Top10	Top20
$c \in \{0:2:10\}$	0.521	0.466	0.626	0.681
$c \in \{0:1:11\}$	0.718	0.671	0.814	0.842
$c \in \{0:0.5:11.5\}$	0.749	0.704	0.834	0.865

많은 보상은 성능의 향상을 가져올 수 있지만, 보다 많은 연산량이 요구된다.

5.4 시스템 수행 시간

제안한 QbSH 시스템은 5.2절에서 언급한 1,000 개의 질의 입력에 대해 약 180분의 처리 시간이 요구되었고, 하나의 입력 질의에 대해서는 평균 10.8초의 처리시간이 필요한 것으로 측정되었다. 이 수치는 거리 함수로 $d^{(\lambda)}_{HINGE}(\cdot)$를 사용하는 경우에 얻어진 수치이다. 상대적으로 $d_{LOG}(\cdot)$ 혹은 $d_{SGI}(\cdot)$를 사용하는 경우는 보다 긴 처리시간이 요구되었다. 본 실험에는 i7-870 프로세서를 갖는 64-bits desktop 컴퓨터가 사용되었으며, 모든 시스템은 C++ 기반으로 구현되었다.

이러한 QbSH 시스템의 수행시간은 제안한 시스템에서 가장 높은 복잡도를 갖는 비대칭적 DTW 거리 차를 구하는 연산에 상당부분 영향을 받는다.

DTW 연산을 위해 $\sum_{i=1}^{I} L(S_q) \times L(S^{(i)}_{DB})$ 와 같은 연산이 요구되기 때문이다. 연산의 복잡도를 줄이기 위해서는 다양한 필터링 관련 연구와 데이터베이스 구조의 최적화 연구가 필요할 것으로 판단된다.

5.5 QbSH 시스템 기반 음원 검색 서비스 구현 결과

제안한 QbSH 시스템을 기반으로 음원 검색 서비스를 구현하였다. 모바일 단말 및 검색 서버 시스템을 기반으로 서비스를 구현한 결과는 그림 5와 같다. 안드로이드 2.2 기반의 모바일 단말은 QbSH 서비스를 위한 클라이언트로 동작한다. 모바일 단말에는 사용자의 허밍 입력에 대해 특성정보를 추출하는 부분, 추출된 특성정보를 서버로 전송하는 부분, 서버에서 제공한 결과를 단말에 출력하는 부분 및 서버와 연결하여 사용자가 원하는 음원을 스트리밍 하여 재생하

그림 5. 모바일 단말 기반의 QbSH 시스템 구현 결과

는 모듈이 포함되어 있다. QbSH 서비스를 위한 서버에는 클라언트 단말이 보내는 데이터를 수신하는 모듈, 비대칭적 DTW기반의 매칭엔진 및 다성음원 기반의 특성정보 데이터베이스가 포함된다.

6. 결 론

디지털 음원의 이용이 이미 아날로그 음원의 이용을 넘어서고, 모바일 서비스 환경의 확대로 인해, 편리하고 효율적인 새로운 검색 시스템에 대한 요구가 증가하고 있다. 이러한 시점에서 사용자의 허밍, 노래 등을 이용한 QbSH 시스템은 새로운 미디어 서비스 환경에 적합한 핵심적인 콘텐츠 검색 기술로 주목을 받고 있으며, 텍스트 입력에 제한이 있는 모바일 기기에 적합한 기술이라 할 수 있다.

지금까지의 대부분의 음원 검색 시스템은 MIDI로 대표되는 단성음원(Monophonic Music)을 기반으로 연구되었다. 이러한 단성음원은 음원의 처리가 비교적 단순하여 검색의 정확성을 확보하고 검색 시간을 줄일 수 있는 장점이 있는 반면, 사용자가 단성음원을 거의 이용하지 않는다는 근본적인 문제가 존재한다.

본 논문은 다성음원 데이터베이스 기반의 효율적인 QbSH 시스템을 제안하였다. 다성음원 및 사용자의 질의 입력에서 추출되는 피치 시퀀스의 부정확성 문제를 사전에 고려하게 매칭엔진을 설계하였으며, 기존 DTW 알고리즘 대신 비대칭 DTW 알고리즘을 매칭엔진에 적용하였다. 또 다양한 거리 함수에 대한 적용을 통해 제안된 QbSH 시스템의 성능을 평가하였다. 450곡, 28시간의 다성음원 데이터베이스를 기반으로 1,000개의 사용자 질의 입력에 대한 성능평가를 통해 제안된 QbSH 시스템의 성능이 MRR 0.718로 확인되었으며, 이는 상용화가 가능한 수준의 검색 성능을 확보한 것으로 판단할 수 있다.

향후 피치 추출에 대한 정확성 확보에 대한 연구, 최적의 거리 함수를 찾을 수 있은 학습 프로세서 및 속도개선을 위한 필터링 프로세서에 대한 추가적인 연구가 필요할 것으로 판단된다.

참 고 문 헌

[1] A. Ghias, J Logan, and D Chamberlin, “Query by Humming : Musical Information Retrieval in an Audio Database,” *Proc. ACM Int. Conf. on Multimedia*, pp. 231-236, 1995.

[2] G. Tzanetakis, “Automatic Genre Classification of Audio Signals,” *IEEE Trans. on Speech and Audio Processing*, Vol.10, No.5, pp. 293-302, 2001.

[3] D. Jang, M. Jin, and C. D. Yoo, “Music Genre Classification using Novel Features and a Weighted Voting Method,” *Proc. Int. Conf. on Multimedia and Expo*, pp. 1377-1380, 2008.

[4] G. Poliner, D. Ellis, A. Ehmann, E. Gomez, S. Streich, and B. Ong, “Melody Transcription from Music Audio: Approaches and Evaluation,” *IEEE Trans. on Audio, Speech, Language Processing*, Vol.15, No.4, pp. 1247-1256, 2007.

[5] S. Jo and C. D. Yoo, “Melody Extraction from Polyphonic Audio Based on Particle Filter,” *Proc. Int. Symp. Music Information Retrieval*, pp. 357-362, 2010.

[6] D. P. W. Ellis and G. E. Poliner, “Identifying Cover Songs with Chroma Features and Dynamic Programming Beat Tracking,” *Proc. Int. Conf Acoustic, Speech and Signal Processing*, Vol.4, pp. 1429-1432, 2007.

[7] S. W. Hainsworth and M. D. Macleod, “Particle Filtering Applied to Musical Tempo Tracking,” *EURASIP Journal on Applied Signal Processing*, Vol.2004, Issue15, pp. 2385-2395, 2004.

[8] J. S. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. D. Yoo, “Audio Fingerprinting Based on Normalized Spectral Subband Moments,” *IEEE Signal Processing Letters*, Vol.13, Issue4, pp. 209-212, 2006.

[9] 허태관, 조황원, 남기표, 이재현, 이석필, 박성주, 박강령, “내용 기반 음원 검출 시스템 구현에 관한 연구,” 한국멀티미디어학회논문지, 제12권, 제11호, pp. 1581-1592, 2009.

[10] H. Sakoe and S. Chiba, “Dynamic Programming Algorithm Optimization for Spoken Word Recognition,” *IEEE Trans. on Acoustics,*

Speech and Signal Processing, Vol.ASSP-26, No.1, pp. 43-49, 1978.

[11] A. Uitdenbogerd and J. Zobel, "Melodic Matching Techniques for Large Music Database," *Proc. ACM Int. Conf. on Multimedia*, pp. 57-66, 1999.

[12] Haus, G. and Pollstri. E, "An Audio Front end for Query-by-Humming Systems," *Proc. Int. Symp. Music Information Retrieval*, pp. 65-72, 2001.

[13] I. Cohen, "Noise Spectrum Estimation in Adverse Environments: Improved Minima Controlled Recursive Averaging," *IEEE Trans. on Speech and Audio Processing*, Vol.11, No.5, pp. 466-475, 2003.

[14] 김기출, 박성주, 이석필, 김무영, "선형 보간법을 이용한 시간과 주파수 조합영역에서의 피치 추정 방법," 전자공학회논문지, 제47권, 제5호, pp. 100-108, 2010.

[15] 윤제열, 이석필, 서경학, 박호종, "하모닉 구조를 이용한 다성 음악의 주요 멜로디 검출," 전자공학회논문지, 제47권, 제5호, pp. 109-116, 2010.

[16] H. M. Yu, W. H. Tsai, and H. M. Wang, "A Query by-Singing System for Retrieving Karaoke Music," *IEEE Trans. on Multimedia*, Vol.10, No.8, pp. 1626-1637, 2008.

[17] J. S. R. Jang and H. R. Lee, "A General Framework of Progressive Filtering and Its Application to Query by Singing/Humming," *IEEE Trans. on Audio, Speech, and Language Processing*, Vol.16, No.2, pp. 350-358, 2008.

[18] Y. Zhu and D. Shasha, "Warping Indexes with Envelope Transforms for Query by Humming," *Proc. Int. Conf. on Management of Data*, pp. 181-192, 2003.

[19] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "On Divergences, Surrogate Loss Functions and Decentralized Detection Department of Statistics," *Tech. Rep. 695*, Dept of Statistics, Univ. of California at Berkeley, 2005.

[20] J. S. R. Jang, N. J. Lee, and C. L. Hsu, "Simple But Effective Methods for QbSH at MIREX 2006," *Proc. Int. Symp. Music Information Retrieval*, pp. 5-7, 2006.

[21] M. Ryynanen and A. Klapuri, "Query by Humming of MIDI and Audio using Locality Sensitive Hashing," *Proc. Int. Conf Acoustic, Speech and Signal Processing*, pp. 2249-2252, 2008.

[22] A. Duda, A. N¨urnberger, and S. Stober, "Towards Query by Humming/Singing on Audio Databases," *Proc. Int. Symp. Music Information Retrieval*, pp. 331-334, 2007.

박 성 주

1995년 2월 경북대학교 전자공학과 학사
1997년 2월 경북대학교 전자공학과 석사
2004년 4월~현재 전자부품연구원 책임연구원

2008년 3월~현재 광운대학교 전자통신공학과 박사과정
관심분야 : 맞춤형 서비스, 유.무선 데이터 스트리밍, IPTV, 오디오 신호 처리, 음원 검색 시스템

정 광 수

1981년 2월 한양대학교 전자 공학과
1983년 2월 한국과학기술원 전기 및 전자공학과 석사
1991년 2월 University of Florida 전기공학과 박사

1983년 3월~1993년 2월 한국전자통신연구원 선임연구원
1993년 3월~현재 광운대학교 전자통신공학과 교수
관심분야 : 인터넷 QoS, 유.무선 비디오 스트리밍, 센서 네트워크