HummingBird: 향상된 스케일드앤워프트 매칭을 이용한 유사 음악 검색 시스템

(HummingBird: A Similar Music Retrieval System using Improved Scaled and Warped Matching)

이 혜 환 † 심 규 석 †† 박 형 민 †††
(Hyehwan Lee) (Kyuseok Shim) (Hyoungmin Park)

요 약 허밍을 통한 유사 검색 질의가 주어질 때 효과적으로 음악 데이타베이스를 검색하는 시스템에 대한 연구는 다양한 방향으로 진행되어 왔다. 최근에는 음악 데이타베이스와 허밍 질의를 시계열 데이타로 변환하여 시계열 데이타의 유사 검색과 관련하여 제안되어 왔던 여러 가지 거리 척도(distance measure)나 인덱싱 기법등을 적용하여 효과적으로 질의를 처리하려는 시도가 계속 되고 있다. 허밍 질의의 특성을 고려하여 균일 스케일링(Uniform Scaling)과 동적 프로그래밍을 사용한 타임 워핑(Dynamic Time Warping)을 함께 고려한 스케일드 앤 워프트 매칭(Scaled and Warped Matching) 거리를 사용하여 효과적인 유사 검색을 하는 방법은 가장 최근 제시된 방법 중 하나이다.

본 논문에서는 허밍을 통한 유사 검색 시스템인 Humming BIRD(Humming Based sImilaR miDi music retrieval system)를 제안하고 구현하였다. 슬라이딩 윈도우를 사용하여 음악의 임의의 부분에 대한 허밍 질의를 처리할 수 있도록 하였으며 더 효율적으로 검색하기 위해 이전의 균일 스케일링을 변형하여 중심을 일치시킨(center-aligned) 균일 스케일링을 제안하고 이와 타임 워핑을 결합한 형태의 스케일드 앤 워프트 매칭을 제안하였다. 이 거리의 좀 더 타이트한 하한을 계산하는 하계 함수를 사용하여 탐색 공간(search space)을 효과적으로 줄여 더 빠르고 효과적인 유사 검색을 가능하도록 하였다. 마지막으로 실험을 통해 개선된 스케일드 앤 워프트 매칭이 이전에 비해 같은 검색 결과를 얻으면서도 효과적으로 검색함을 탐색 공간을 줄이는 가지치기 성능을 비교함으로써 보였다.

키워드 : Humming BIRD, 유사 음악 검색 시스템, 허밍 질의, 시계열 데이타, 스케일드 앤 워프트 매칭

Abstract Database community focuses on the similar music retrieval systems for music database when a humming query is given. One of the approaches is converting the midi data to time series, building their indices and performing the similarity search on them. Queries based on humming can be transformed to time series by using the known pitch detection algorithms. The recently suggested algorithm, scaled and warped matching, is based on dynamic time warping and uniform scaling.

This paper proposes Humming BIRD(Humming Based sImilaR miDi music retrieval system) using sliding window and center-aligned scaled and warped matching. Center-aligned scaled and warped matching is a mixed distance measure of center-aligned uniform scaling and time warping. The newly proposed measure gives tighter lower bound than previous ones which results in reduced search space. The empirical results show the superiority of this algorithm comparing the pruning power while it returns the same results.

Key words : Humming BIRD, similar music retrieval system, time series data, similarity search, scaled and warped matching

· 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 · 지원 사업(IITA-2006-C1090-0603-0031)의 연구결과로 수행되었음

† 학생회원 : 서울대학교 전기컴퓨터공학부
hhlee@kdd.snu.ac.kr
†† 정 회 원 : 서울대학교 전기컴퓨터공학부 교수
shim@ee.snu.ac.kr
††† 학생회원 : 서울대학교 전기컴퓨터공학부
hmpark@kdd.snu.ac.kr

논문접수 : 2007년 1월 22일
심사완료 : 2007년 6월 28일

1. 서 론

기존의 널리 사용되어 오던 유사 음악 검색 시스템들은 대부분 키워드 기반의 검색 시스템이었기 때문에 사용자가 제목이나 가수 혹은 가사와 같은 음악에 관한 텍스트 형태의 메타 정보를 갖지 못하고 멜로디로만 어

렴풋이 기억하는 음악은 검색할 수가 없었다. 실제로 누구나 한번쯤은 제목, 가수, 가사 등의 정보를 알지 못한 채 머리 속에 맴도는 멜로디만으로 노래를 검색할 수 없어 답답했던 경험이 있을 것이다. 이를 해결하기 위하여 연구되고 있는 것이 내용에 기반한 질의 시스템인 허밍 질의를 통한 유사 음악 검색 시스템이다. 허밍 질의를 통한 유사 음악 검색 시스템의 사용자는 자신이 기억하는 노래의 일부를 허밍하고 시스템은 허밍한 부분과 비슷한 부분을 포함하는 음악의 목록을 검색하여 보여준다.

휴대폰에서 사용할 수 있는 멀티미디어 콘텐츠와 서비스가 다양해지고 mp3와 같은 기능도 겸할 수 있게 되면서 통신 업계에서는 휴대폰을 통해 사용자로부터 허밍 입력을 받아 원하는 음악을 검색하고 들을 수 있는 서비스를 제공하기 시작했다. SK 텔레콤에서는 디지털 콘텐츠 검색 기술 전문업체인 나요 미디어와 함께 전화를 통해 노래의 일부분을 허밍하면 이와 유사한 음악을 찾아주는 허밍 기반 음악 검색 엔진을 개발, 세계 최초로 상용화하여 서비스를 시작하였다. 나요 미디어는 인터넷을 통해서도 사용자가 찾는 노래의 일부를 허밍하면 유사한 음악을 검색해주는 서비스를 제공하고 있다. 일본의 NTT도코모, 도시바 등에서도 이와 관련된 기술 개발이 이루어지면서 허밍 질의를 통한 유사 음악 검색 시스템은 본격적으로 모바일 음악 서비스에 적용될 예정이다. 이러한 서비스를 통해 휴대폰을 통한 서비스를 사용하여 벨소리, 통화 연결음, MP3 파일등을 사용자가 손쉽게 검색, 다운로드 받을 수 있을 뿐 아니라 이를 활용한 부가 서비스 개발에도 기여할 수 있을 것으로 기대된다. 이외에도 음반 매장이나 노래방과 같은 곳에서 원하는 노래를 찾고 싶은데 제목이나 가수의 정보를 기억하지 못해 고민했던 사용자들에게 허밍 질의를 통한 유사 음악 검색 시스템은 유용하게 사용될 것이다. 또한 작곡가들에게는 자신이 작곡한 멜로디의 창작성 여부를 검증해 보는데 사용될 수도 있다.

이러한 추세와 맞물려 허밍 질의를 통한 유사 음악 검색 시스템에 대한 관심이 높아지고 연구가 활발히 이루어지고 있다. 국내외 대학 및 연구소에서 허밍 질의를 처리하는 유사 음악 검색 시스템에 대한 다양한 프로젝트를 진행하며 [1,2]등의 데모 버전을 제공하고 있으며, [3,4]와 같은 특허들도 출원되고 있다. 그러나 현재까지의 허밍을 통한 유사 음악 검색 시스템은 대용량 음악 데이타에서 다양한 사용자의 허밍에 대해 적절한 결과를 돌려주지 못하는 경우가 발생하고 검색 시간이 오래 걸리는 등 아직 많은 문제점을 가지고 있다.

본 논문에서는 허밍 질의와 미디 데이타를 시계열 데이타로 변환하여 시계열 데이타의 유사성을 검색하는 유사 음악 검색 시스템인 Humming BIRD를 구현하였다. 임의의 부분에 대한 허밍 질의를 처리하여 원하는 유사 검색 결과를 얻기 위해 미디 데이타를 슬라이딩 윈도우를 사용하여 인덱싱 하였다. 또한 허밍 질의에 적합한 적절한 거리 함수를 선택하고 개선하여 효율적으로 검색을 할 수 있는 방법을 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 허밍 질의의 통한 유사 음악 검색 시스템들의 접근 방법을 소개한다. 3장에서는 본 논문이 바탕을 두고 있는 거리 척도인 균일 스케일링, 타임 워핑, 스케일드 앤 워프트 매칭에 대해 설명하고 효율적인 유사 검색에 사용될 수 있는 각 거리 척도의 하계 함수(lower bounding function)를 설명한다. 4장에서는 전체적인 Humming BIRD 시스템의 개요를 설명하고 5장에서는 새롭게 제시한 중심을 일치시킨 균일 스케일링, 이를 이용한 중심을 일치시킨 스케일드 앤 워프트 매칭과 각각에 대한 하계 함수를 정의하고 설명한다. 마지막으로 6장에서는 기존의 스케일드 앤 워프트 매칭을 사용한 것과 성능을 비교한다.

2. 관련 연구

허밍을 이용한 유사 음악 검색 시스템은 멀티미디어 데이타베이스 분야에서 다양한 방향으로 연구되어 오고 있다. 몇몇 시스템[5-8]이 이미 제시되었으며 그 중 중요한 몇 가지 접근 방법을 소개하겠다.

이전의 많은 허밍 질의 처리 시스템들에서는 사용자가 원하는 음악을 찾기 위해서 음조를 사용하였다. 음조란 음의 상대적인 높이의 변화를 뜻하는 것으로서 사용자의 허밍 질의는 분리된 음표의 시퀀스로 변환되고 이로부터 음조가 얻어진다. 음조는 보통 몇 개의 알파벳으로 나타내어진다[9]. 데이타베이스의 음악도 마찬가지 방법으로 음조로 표현한다. 이렇게 하면 멜로디는 몇 개의 알파벳으로 된 문자열로 표현되고, 편집 거리(Edit distance)를 사용하여 유사한 멜로디의 음악을 검색할 수 있다. 이 때 문자열 매칭 분야의 q-gram과 같은 기법을 사용하여 좋은 성능을 얻을 수 있다[10,11].

음조를 이용한 방법은 사용자의 허밍이 정확하지 않은 경우에도 멜로디를 잘 추출할 수 있다는 점에서 유리하다. 그러나 음표의 길이 등을 무시하고 이전 음표에 대비한 상대적인 높낮이만으로 멜로디 정보를 표현하기 때문에 정확한 유사 검색의 결과를 얻기가 힘들다. 음조에 기반한 유사 검색을 수행하는 허밍 시스템에서 허밍 질의로부터 분리된 음표를 추출하는 안정적인 알고리즘이 존재하지 않아 원하는 검색 결과를 얻기 힘들었기 때문에 질의로부터 음표를 분리하는 과정을 거치지 않으려는 연구가 있었다[12]. 이 방법은 이전의 음조에 기

반한 시스템에 비해 훨씬 좋은 결과를 얻었지만 시간이 오래 걸려서 실제로 허밍 시스템에서 사용하기 힘들다.

한편, 시계열 데이타베이스에서의 유사 검색은 데이타 마이닝 분야에서 활발히 연구돼 왔던 분야이다. 여러 거리 함수와 효율적인 유사 검색을 위한 기법들이 이미 연구되어 있어서 허밍 질의와 미디 데이타베이스를 시계열 데이타와 같이 볼 수 있다면 기존의 기법들을 사용할 수 있을 것이라고 여겨졌고 이와 관련된 연구가 진행 되어 왔다. [13]에서는 시계열 데이타베이스에서 사용되는 유사 검색 기법을 이용한 허밍 질의 처리 시스템을 구현하였다. 이 때 두 시계열 데이타간의 유사성을 측정하기 위해 타임 워핑 거리를 사용하였다.

[14,15]에서는 허밍질의 시스템에 확장, 적용 시킬 수 있는 거리 척도인 균일 스케일링 거리와 스케일드 앤 워프트 매칭 거리를 정의하고 각각의 하한을 계산하는 하계 거리 함수를 제시하였다. 균일 스케일링 거리 척도는 허밍 질의를 할 때에 사용자가 전체적인 빠르기를 빠르게 혹은 느리게 하는 경우에도 원하는 유사 검색 결과를 얻을 수 있도록 한다. 스케일드 앤 워프트 매칭 거리 함수는 균일 스케일링과 타임 워핑 거리를 결합한 형태이다.

3. 기본 개념

이 장에서는 시계열 데이타의 유사 검색에 사용되어 왔던 기존의 여러 거리 척도 중 유사 음악 검색 시스템과 관련이 있는 균일 스케일링 거리, 타임 워핑 거리, 스케일드 앤 워프트 매칭 거리의 정의와 이를 사용하여 효율적으로 유사 검색을 수행하기 위한 하계 함수를 살펴본다.

3.1 균일 스케일링 거리

균일 스케일링 거리는 두 시퀀스의 거리를 계산할 때 시간 축 상으로의 전체적인 확대와 축소를 고려하는 거리 척도이다. 허밍 질의와 같이 질의 시퀀스가 전체적으로 빠르게 혹은 느리게 생성될 수 있을 때에 유용하다.

균일 스케일링 거리를 사용하는 유사 검색의 목표는 질의 시퀀스 Q와 데이타베이스의 시퀀스 C, 최대 스케일링 상수 ℓ이 주어질 때, 데이타를 최대 ℓ배, 최소 $1/\ell$배로 균일하게 확대하여 Q와 가장 작은 유클리드 거리를 갖는 C의 접두 서브 시퀀스를 찾는 것이다.

정의 3.1 [균일 스케일링 거리] 질의 시퀀스 $Q=[q_1, \cdots, q_m]$, 데이타베이스의 시퀀스 $C=[c_1, \cdots, c_n](m \le n)$, 최대 스케일링 상수 $\ell(\ell \ge 1, n \ge \ell m)$ 이 주어졌을 때, $\lceil m/\ell \rceil \le q \le \lceil m\ell \rceil$ 인 q에 대하여 C(m, q)는 길이 q인 C의 접두 서브 시퀀스 $[c_1, \cdots, c_q]$가 질의 시퀀스 Q의 길이 m과 같아지도록 스케일링된 것이다. 즉, $C(m,q)$의 i($1 \le i \le m$)번째 요소 $C(m,q)_i$에 대해 $C(m,q)_i = c_{iq/m\rfloor}$이 성립한다.

그러면, C와 Q 사이의 균일 스케일링 거리 $US(C,Q,l)$은 다음과 같이 정의 된다. 이 때, D는 유클리드 거리이다.

$$US(C,Q,l) = \min_{\lfloor m/\ell \rfloor \le q \le \lfloor \ell m \rfloor} D(C(m,q), Q)$$

가능한 각각의 q에 대하여 유클리드 거리를 구하는데 O(m)의 시간이 걸리므로 이 거리를 구하는 데 걸리는 시간 복잡도는 $O(m^2)$이다.

```
NNSearch(Q)
begin
MinDist := ∞
TrueDist := ∞
tmp := ∞
IndexOfBest := 1
for each Ci in sequence database do {
    tmp = D(Qlb, Ci)
    if tmp<MinDist {
        TrueDist = D(Q, Ci)
        if TrueDist<MinDist {
            MinDist = TrueDist
            IndexOfBest = I
        }
    }
}
return (MinDist, IndexOfBest)
end
```

그림 1 하계 거리 함수를 사용한 NNSearch

그림 1은 하계 함수를 사용하여 질의와 가장 유사한 시퀀스를 찾는 알고리즘이다. 데이타베이스의 각 시퀀스에 대하여 CPU 비용이 큰 실제 거리를 계산하기에 앞서서 알고리즘의 세번째 줄에서 실제 거리의 하한을 계산하는 하계 함수를 먼저 계산한다. 다섯번째 줄부터 열 세번째 줄에서 그 거리를 사용하여 가장 가까운 시퀀스가 될 가능성이 있는 데이타에 대해서만 실제 거리를 계산한다. 시퀀스의 차원이 수백 차원 정도까지 커질 수 있다는 점을 생각할 때 모든 데이타에 대해 질의 시퀀스와의 균일 스케일링 거리를 계산하는 것은 상당한 시간을 필요로 한다. 따라서 빠른 시간 내에 계산할 수 있는 실제 거리의 하한을 계산하는 하계 함수를 정의하고 이 하계 함수를 사용하여 가지치기 하는 기법을 보편적으로 사용한다.

균일 스케일링 거리의 하계 함수를 정의하기 위한 데이타 시퀀스의 엔벨로프(envelope)를 다음과 같이 정의한다.

정의 3.2 [균일 스케일링에서의 엔벨로프] 질의 시퀀스 $Q=[q_1, \cdots, q_m]$, 데이타베이스의 시퀀스 $C=[c_1 \cdots, c_n](m \le n)$, 최대 스케일링 상수 $\ell(\ell \ge 1, n \ge \ell m)$ 이 주어졌을 때, 데이타 시퀀스 C의 엔벨로프는 두 시퀀스 $UC=[uc_1, \cdots, uc_m]$와 $LC=[lc_1, \cdots, lc_m]$으로 이루어지며, $1 \le i \le m$인 i에 대하여 각 요소는 다음과 같다.

$$uc_i = \max(c_{i/l\rfloor}, \ldots, c_{il\rfloor}) \ , \ lc_i = \min(c_{i/l\rfloor}, \ldots, c_{il\rfloor})$$

Q, C사이의 US(Q, C, ℓ)의 하계 함수 정의는 다음과 같다.

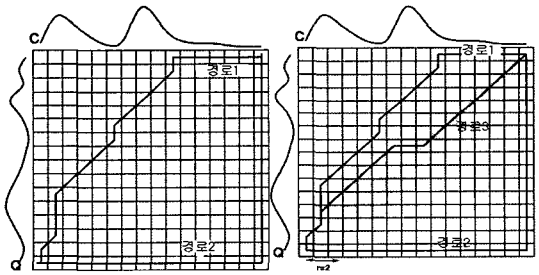
정의 3.3 [하계 함수] 두 시퀀스 Q= [q_1, …, q_m], C= [c_1, …, c_n]와 C의 엔벨로프 U=[u_1, …, u_m]와 L=[l_1, …, l_m]이 주어졌을 때, 하계 함수 $LB(Q,C)$는 다음과 같이 정의된다.

$$LB(Q,C)=\sum_{i=1}^{m}\begin{cases}(q_i-uc_i)^2 \text{ if } q_i > uc_i\\(q_i-lc_i)^2 \text{ if } q_i < lc_i\\0 \quad otherwise\end{cases}$$

타임 워핑 거리와 스케일드 앤 워프트 매칭에서도 엔벨로프를 정의한 후 정의 3.3의 하계 함수를 사용한다.

3.2 타임 워핑 거리

타임 워핑은 두 시퀀스 간의 각 요소를 매칭할 때, 요소 값을 임의의 수만큼 반복하는 것을 허용하는 변환이다. 유클리드 거리와 같은 L_p-norm 계열의 경우 시간축이 조금이라도 어긋나면 시퀀스 간의 거리가 매우 커지는데 반해 타임 워핑 거리는 시퀀스 내의 각 요소가 반복되는 것을 허용하여 두 시퀀스를 축의 뒤틀림을 보정하여 거리를 계산한다.



(a) 타임 워핑 경로 (b)제한된 타임워핑 경로

그림 2 타임 워핑 경로와 제한된 타임 워핑 경로

그림 2(a)는 두 시퀀스 Q와 C간의 타임 워핑 거리를 계산할 때 생길 수 있는 매칭 경로를 보여준다. 경로 1의 경우 Q의 마지막 한 요소가 C의 요소 7개와 매칭된다. 극단적인 경우 경로 2에서와 같이 Q의 처음 한 요소가 C의 모든 16개의 요소와 매칭되는 경우도 생길 수 있다. 이는 어느 정도 시간상의 뒤틀림(warping)을 허용하자는 타임 워핑 거리의 취지와 맞지 않으므로 이를 막기 위하여 타임 워핑 제한 상수 r을 갖는 제한된 타임 워핑 거리를 사용한다. 그림 2(b)에서는 어둡게 표시된 부분만을 경로가 통과할 수 있도록 제한(r=2)을 주었다. 타임 워핑 제한 상수 r은 타임 워핑 경로가 대각선으로부터 얼마나 벗어날 수 있는지를 제한하는 값으로, 그림 2(b)에서는 대각선 양쪽으로 2칸씩의 벗어남을 허용하였다. 이 경우 경로 1,2와 같은 매칭 경로는 생길 수가 없고 경로 3만이 유효하다. 타임 워핑 제한 상수 r을 갖는 제한된 타임 워핑 거리를 다음과 같이 정의한다.

정의 3.4 [제한된 타임 워핑 거리] 두 시퀀스 Q= [q_1, …, q_m], C= [c_1, …, c_n], 타임 워핑 제한 상수 r이 주어졌을 때, 타임 워핑 거리 $cDTW(C,Q,r)$은 다음과 같이 재귀적으로 정의된다.

$$D_r=\begin{cases}(c_i-q_i)^2 \text{ if } |i-j|\le r\\\infty \quad otherwise\end{cases}$$
$$cDTW(\phi,\phi,r)=0$$
$$cDTW(C,\phi,r)=cDTW(\phi,Q,r)=\infty$$
$$cDTW(C,Q,r)=D_r(First(C),First(Q))+\min\begin{cases}cDTW(C,Rest(Q),r)\\cDTW(Rest(C),Q,r)\\cDTW(Rest(C),Rest(Q),r)\end{cases}$$

이 때, ϕ는 비어있는 시퀀스, $First(C)$는 C의 첫번째 값 c_1, $Rest(C)$는 C에서 c_1을 뺀 시퀀스 [c_2, …, c_n]이다.

타임 워핑 거리의 하한을 계산하는 하계 함수를 정의하기 위해 사용하는 엔벨로프는 다음과 같이 정의된다.

정의 3.5 [타임 워핑에서의 엔벨로프] 질의 시퀀스 Q= [q_1, …, q_m], 데이타베이스의 시퀀스 C= [c_1, …, c_n], 타임 워핑 제한 상수 r이 주어질 때, C의 엔벨로프는 UW=[uw_1, …, uw_m]와 LW=[lw_1, …, lw_m]로 이뤄지고, $1\le i\le m$인 i에 대해 각 요소는 다음과 같다.

$$uw_i=\max(c_{\max(1,i-r)},\ldots,c_{\min(i+r,n)}),$$
$$lw_i=\min(c_{\max(1,i-r)},\ldots,c_{\min(i+r,n)})$$

3.3 스케일드 앤 워프트 매칭

허밍을 통한 유사 음악 검색 시스템에서 요구되는 거리 함수는 균일 스케일링 거리 함수와 타임 워핑 거리 함수의 특성을 모두 갖추고 있어야 한다. [14]에서는 두 거리 함수를 결합한 스케일드 앤 워프트 매칭을 제안하고 있다.

정의 3.6 [스케일드 앤 워프트 매칭 거리] 질의 시퀀스 Q= [q_1, …, q_m], 데이타베이스의 시퀀스 C= [c_1, …, c_n](m≤n), 최대 스케일링 상수 ℓ ($\ell\ge1$, $n\ge\ell m$), 타임 워핑 제한 상수 r 이 주어졌을 때, $\lceil m/\ell\rceil\le q\le\lceil m\ell\rceil$에 대하여 $C(m,q)$는 정의 3.1에서와 마찬가지로 길이가 q인 C의 접두 서브 시퀀스를 길이 m이 되도록 스케일링한 것이다. C와 Q 사이의 스케일드 앤 워프트 매칭 거리 $SWM(C,Q,l,r)$은 다음과 같다. 이 때, cDTW는 제한된 타임 워핑 거리 함수이다.

$$SWM(C,Q,l,r)=\min_{|m/l|\le q\le|lm|}cDTW(C(m,q),Q,r)$$

균일 스케일링과 타임 워핑을 함께 적용할 때 타임 워핑 제한 상수를 주의하여야 한다. 일반적인 경우 타임 워핑 제한 상수는 전체 데이타 시퀀스에서 타임 워핑을 허용하는 비율로 주어진다. 예를 들어 워핑 허용 비율이 0.05라면 r=0.05|C|로 정해진다. 스케일드 앤 워프트 매칭에서는 데이타의 서브 시퀀스를 스케일링 한 후에 타임 워핑 거리를 적용하므로 스케일링 상수가 ρ라면 균

일 스케일링 한 후 시퀀스의 길이는 |ρC|가 되고 따라서 r=0.05|ρC|로 정해진다. r'을 스케일링을 적용하지 않았을 때의 타임 워핑 제한 상수라 할 때 r = 0.05|ρC|= ρ|0.05 C|= ρr'와 같이 표현된다. 이후부터 타임 워핑 제한 상수가 r일 때, 스케일링을 적용하지 않은 타임 워핑 제한 상수를 r'으로 표현하겠다. 스케일드 앤 워프트 매칭 거리의 엔벨로프의 정의는 다음과 같다.

정의 3.7 [스케일드 앤 워프트 매칭에서의 엔벨로프] 질의 시퀀스 $Q=[q_1, \cdots, q_m]$, 데이타베이스의 시퀀스 $C=[c_1, \cdots, c_n]$ $(m \le n)$, 최대 스케일링 상수 ℓ ($\ell \ge 1$, $n \ge \ell m$), 타임 워핑 제한 상수 r이 주어질 때, C의 엔벨로프는 $U=[u_1, \cdots, u_m]$과 $L=[l_1, \cdots, l_m]$로 구성되고, 각 요소$(1 \le i \le m)$는 다음과 같다.

$$u_i = \max(uw_{\lfloor i/\ell \rfloor}, \ldots, uw_{\lceil i\ell \rceil}) = \max(c_{\lfloor i/\ell \rfloor - r'}, \ldots, c_{\lceil i\ell \rceil + r'}),$$
$$l_i = \min(lw_{\lfloor i/\ell \rfloor}, \ldots, lw_{\lceil i\ell \rceil}) = \min(c_{\lfloor i/\ell \rfloor - r'}, \ldots, c_{\lceil i\ell \rceil + r'})$$

4. Humming BIRD 시스템의 개요

본 논문에서 제안하는 허밍 질의 처리 시스템 Humming BIRD는 그림 3과 같이 미디 데이타 처리 모듈, 허밍 입력 처리 모듈. 유사 검색 수행 모듈들로 구성된다.

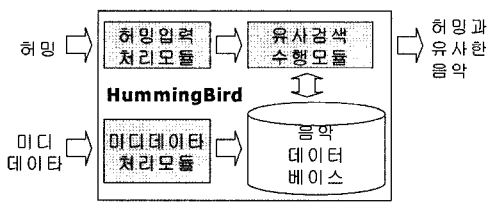


그림 3 HummingBird 시스템

미디 데이타 처리 모듈은 미디 데이타를 전처리하여 시계열 데이타를 얻는다. 미디 파일은 악보의 모든 내용을 이벤트로 표현하는데 우리가 관심이 있는 멜로디는 Note On, Note Off 라는 이벤트들로 표현이 된다. 이로부터 음정과 지속시간을 나타내는 (Note, Duration)과 같은 2차원 벡터형태의 멜로디를 추출할 수 있다. 이를 'n, n, ..., n' 과 같이 음의 시퀀스로 변환하고 헤더 정보를 이용하여 한 마디가 일정한 개수의 값으로 나타나도록 샘플링한다. 사용자가 음에 비해서 쉼표를 정확하게 지키기가 힘들고 허밍으로부터 쉼표를 정확히 추출하는 것 또한 힘들기 때문에 미디 파일의 처리에서도 쉼표를 무시하고 이전 음이 계속되는 것으로 표현한다. 이 시스템에서는 사용자가 일정 수 내의 마디를 허밍한다고 가정하고 슬라이딩 윈도우를 사용한다. 이 경우 사용자가 노래의 임의의 부분을 허밍하는 경우에도 일치하는 서브 시퀀스를 모두 찾을 수 있다.

허밍 입력 처리 모듈은 사용자의 허밍 입력을 처리한다. 사용자가 마이크를 사용하여 허밍하면 이를 10ms 단위의 프레임으로 쪼개고 각 프레임 당 피치 추출 알고리즘을 적용하여 피치를 추출하여 시계열 데이타로 변환한다.

유사 검색 수행 모듈은 허밍 질의와 유사한 음악을 찾기 위한 유사 검색을 수행한다. 의미 있는 결과를 얻기 위해 유사 검색에 앞서 허밍 질의와 음악 데이타를 정규화하며 전체적인 빠르기나 부분적인 빠르기의 변화도 고려하면서도 효율적으로 검색 하도록 균일 스케일링을 개선하여 적용하였다. 미디 파일을 처리한 결과로 얻어지는 시퀀스의 슬라이딩 윈도우로부터 차원 감소 기법을 사용하여 피처(feature)를 추출하고 이를 사용하여 R*-tree와 같은 다차원 인덱스를 구축한다. 질의 시퀀스로부터도 피처를 추출하고, 인덱스 구조를 사용하여 실제 유사한 데이타일 가능성이 있는 데이타만 실제 데이타를 디스크로부터 읽고 실제 비교를 수행한다.

5. 향상된 스케일드 앤 워프트 매칭을 이용한 유사 검색

좀 더 효과적으로 가지치기를 하고자 기존의 균일 스케일링 거리를 변형하여 중심을 일치시킨 균일 스케일링 거리를 제안하고 이를 사용한 개선된 형태의 스케일드 앤 워프트 매칭 거리를 제안하고 설명한다. 또한 각각의 거리 척도를 사용하여 효율적으로 유사 검색을 수행하고자 사용하는 하계 함수를 정의하고 실제 거리의 하한을 계산함을 보인다.

5.1 중심을 일치시킨 균일 스케일링 거리

실제 거리의 하한을 계산하는 하계 함수를 사용하여 그림 1의 알고리즘과 같은 방식으로 가지치기를 하는 것이 효과적이기 위해서는 하계 함수가 실제 거리의 하한에 최대한 가깝도록 계산하여야 한다. 만약 그 하한이 너무 느슨한 경우에는 가지치기를 하는 것의 효과가 거의 없다. 간단한 예로 항상 0의 값을 갖는 하계 함수를 가정하자. 이 경우 0은 0이상인 어떤 거리 값보다도 작거나 같으므로 모든 거리 척도에 대해 하한이다. 하지만 그림 1의 네번째 줄에서 모든 C_i에 대하여 tmp 값이 0이므로 전혀 가지치기를 하지 못하고 모든 데이타 시퀀스에 대하여 실제 거리를 계산해야 한다. 따라서 하계 함수를 타이트하게 정의 하는 것은 가지치기 성능을 좌우하여 효율적인 유사 검색을 가능하게 한다. 기존의 타임 워핑 거리에 관해서도 이와 같이 하계 함수를 좀 더 타이트하게 하기 위한 시도가 있어왔다. 본 논문도 이러한 관점에서 효율적으로 유사 검색을 수행하고자 더 타이트한 하계 함수와 이를 위해 실제 데이타를 더욱 타이트하게 감싸는 엔벨로프를 제안한다.

예 5.1 8차원 질의 시퀀스 $Q=[q_1, \cdots, q_8]$와 16차원

데이타 시퀀스 C=[c_1, ⋯, c_{16}]가 주어지고 ℓ은 2일 때 기존의 균일 스케일링에서의 엔벨로프는 아래와 같이 정해진다.

uc_1= max(c_1, c_2) lc_1= min(c_1, c_2)
⋯ ⋯
uc_4= max(c_2,c_3,⋯,c_7,c_8) lc_4= min(c_2,c_3,⋯,c_7,c_8)
uc_5= max(c_3,c_4,⋯,c_9,c_{10}) lc_5= min(c_3,c_4,⋯,c_9,c_{10})
⋯ ⋯
uc_8= max(c_4,c_5,⋯,c_{15},c_{16}) lc_8= min(c_4,c_5,⋯,c_{15},c_{16})

예 5.1과 같이 기존의 엔벨로프는 각 차원 i에 대해 $c_{\lceil i/\ell \rceil}$부터 $c_{\lceil i\ell \rceil}$ 사이의 최대값과 최소값을 uc_i와 lc_i로 정하므로 i가 커질수록 uc_i와 lc_i를 정하기 위해서 봐야 하는 범위가 커지고 따라서 엔벨로프가 느슨해진다. 본 논문에서는 이를 개선하기 위하여 스케일링을 할 때에 시퀀스의 중심을 고정시키고 양쪽으로 균일하게 늘리는, 즉 중심을 일치시킨 균일 스케일링을 제안한다. 중심을 일치시킨 균일 스케일링에서는 C'(m,q)을 C와 중심을 일치 시킨 길이가 q인 서브 시퀀스를 길이 m이 되도록 균일하게 스케일링하여 얻은 시퀀스로 정한다.

예 5.2 길이 8인 시퀀스 C=[c_1, c_2, ..., c_7, c_8]이 주어졌을 때 길이 q($1 \le q \le 8$)에 따른 C와 중심을 일치시킨 서브 시퀀스와 이를 길이 4가 되도록 스케일링한 C(4,q)는 다음과 같다.

q	중심을 일치 시킨 서브 시퀀스	C(4,q)
1	[c_5]	[c_5, c_5, c_5, c_5]
2	[c_4, c_5]	[c_4, c_4, c_5, c_5]
⋯	⋯	⋯
7	[c_2, c_3, c_4, c_5, c_6, c_7, c_8]	[c_3, c_5, c_7, c_8]
8	[c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8]	[c_2, c_4, c_6, c_8]

예 5.2와 같이 C와 중심을 일치시킨 C의 서브 시퀀스를 질의 시퀀스의 길이 m과 같도록 스케일링 하여 유클리드 거리를 구한 것을 중심을 일치시킨 균일 스케일링 거리로 정의한다.

정의 5.1 [중심을 일치시킨 균일 스케일링 거리] 질의 시퀀스 Q=[q_1,⋯, q_m], 데이타베이스의 시퀀스 C=[c_1, ⋯, c_n](m≤n), 최대 스케일링 상수 ℓ($\ell \ge 1$, $n \ge \ell m$)이 주어졌을 때, $\lceil m/\ell \rceil \le q \le \lceil m\ell \rceil$인 q에 대하여 C'(m, q)는 길이가 q이며 C와 중심을 일치시킨 C의 서브 시퀀스를 질의 시퀀스의 길이 m과 같아지도록 균일 스케일링 한 것이다. 즉, $C'(m,q)$의 i($1 \le i \le m$)번째 요소 $C'(m,q)_i$는 다음과 같이 표현된다.

$$C'(m,q)_i = c_{(n-q)/2+\lfloor iq/m \rfloor}$$

이 때 C, Q 간의 중심을 일치시킨 균일 스케일링 거리 $US_c(C,Q,l)$은 다음과 같다. D는 유클리드 거리이다.

$$US_c(C,Q,l) = \min_{\lceil m/l \rceil \le q \le \lfloor lm \rfloor} D(C'(m,q), Q)$$

중심을 일치시킨 균일 스케일링 거리를 사용하더라도 Humming BIRD와 같이 슬라이딩 윈도우를 적용하는 경우, 기존의 균일 스케일링 거리를 사용하는 것과 동일한 결과를 얻는다. 본 논문에서는 편의상 슬라이딩 윈도우가 데이타 포인트 하나씩 옮겨 가는 경우로 설명한다. 실제 구현한 시스템에서는 슬라이딩 윈도우를 한 마디씩 옮겨 가는데 이 경우에도 기존의 균일 스케일링 거리를 사용하는 경우와 새롭게 정의된 스케일링 거리를 사용하는 경우가 같은 탐색 공간을 뒤지므로 동일한 검색 결과를 얻는다. 중심을 일치시킨 균일 스케일링 거리의 엔벨로프는 다음과 같이 정의한다.

정의 5.2 [중심을 일치시킨 균일 스케일링에서의 엔벨로프] 질의 시퀀스 Q= [q_1, ⋯, q_m], 데이타베이스의 시퀀스 C= [c_1, ⋯, c_n](m≤ n), 최대 스케일링 상수 ℓ ($\ell \ge 1$, $n \ge \ell m$) 이 주어졌을 때, 데이타 시퀀스 C의 엔벨로프는 두 시퀀스 UC=[uc_1, ⋯, uc_m]와 LC=[lc_1, ⋯, lc_m]으로 이루어지며, $1 \le i \le m$인 i에 대하여 각 요소는 다음과 같다.

$$u_i = max(c_{from}, ..., c_{to}), \qquad l_i = min(c_{from}, ..., c_{to})$$

$$from = \begin{cases} max\left(\left\lceil \frac{n}{2} + ml\left(\frac{i}{m} - \frac{1}{2}\right)\right\rceil - r', 1\right) & \text{if } i \le \frac{m}{2} \\ max\left(\left\lceil \frac{n}{2} + \frac{m}{l}\left(\frac{i}{m} - \frac{1}{2}\right)\right\rceil - r', 1\right) & \text{if } i > \frac{m}{2} \end{cases}$$

$$to = \begin{cases} min\left(\left\lfloor \frac{n+3}{2} + \frac{m}{l}\left(\frac{i}{m} - \frac{1}{2}\right)\right\rfloor + r', n\right) & \text{if } i \le \frac{m}{2} \\ min\left(\left\lfloor \frac{n+3}{2} + ml\left(\frac{i}{m} - \frac{1}{2}\right)\right\rfloor + r', n\right) & \text{if } i > \frac{m}{2} \end{cases}$$

예 5.3 예 5.1에서와 같이 경우에 중심을 일치시킨 균일 스케일링에 대한 엔벨로프는 다음과 같이 정해진다.

uc_1=max(c_2, c_3, ⋯, c_7, c_8) lc_1=min(c_2, c_3, ⋯, c_7, c_8)
⋯ ⋯
uc_4=max(c_8, c_9) lc_4=min(c_8, c_9)
uc_5=max(c_9, c_{10}, c_{11}) lc_5=min(c_9, c_{10}, c_{11})
⋯ ⋯
uc_8=max(c_{10}, c_{11}, ⋯, c_{15}, c_{16}) lc_8=min(c_{10}, c_{11}, ⋯, c_{15}, c_{16})

새로운 균일 스케일링의 엔벨로프는 중심에서 타이트하고 양 끝으로 갈수록 느슨해진다. 예에서도 확인 할 수 있듯이 기존의 경우 가장 느슨한 경우 i=8일 때 c_4부터 c_{16}까지 13개의 값을 보아야 했던 것에 비해 새로운 엔벨로프의 경우 가장 느슨한 경우 i=1 또는 8일 때 c_2부터 c_8까지 혹은 c_{10}부터 c_{16} 까지 7개의 값을 보고 그 중 최대, 최소 값이 엔벨로프를 정한다. 새로운 엔벨로프가 기존의 엔벨로프보다 타이트함을 보이기 위해 m차원의 엔벨로프를 구하기 위해 각 차원에서 봐야하는 범위의 합을 비교해 보겠다.

기존의 균일 스케일링 거리의 엔벨로프 i번째 값은 $c_{\lceil i/\ell \rceil}$부터 $c_{\lceil i\ell \rceil}$까지 중에서 최소값과 최대값이므로 전체 엔벨로프를 구하기 위해 보아야 하는 범위의 합은 다음과 같다.

$$l_{old} = \sum_{i=1}^{m} (\lceil il \rceil - \left\lfloor \frac{i}{l} \right\rfloor + 1) \ge \sum_{i=1}^{m} (il - \frac{i}{l}) = \frac{m(m+1)}{2}(l - \frac{1}{l})$$

마찬가지로 정의 5.2에 따라 중심을 일치시킨 균일 스케일링 거리의 전체 엔벨로프를 구하기 위해 보아야 하는 범위의 합은 다음과 같이 계산된다.

$$l_{\neq w} = \sum_{i=1}^{\lfloor m/2 \rfloor} \left(\left|\frac{n+3}{2}+\frac{m}{l}\left(\frac{i}{m}-\frac{1}{2}\right)\right| - \left|\frac{n}{2}+lm\left(\frac{i}{m}-\frac{1}{2}\right)\right| + 1\right)$$
$$+ \sum_{i=\lfloor \frac{m}{2} \rfloor + 1}^{m} \left(\left|\frac{n+3}{2}+ml\left(\frac{i}{m}-\frac{1}{2}\right)\right| - \left|\frac{n}{2}+\frac{l}{m}\left(\frac{i}{m}-\frac{1}{2}\right)\right| + 1\right)$$
$$\leq \sum_{i=1}^{\lfloor m/2 \rfloor} \left(\frac{1}{l}-l\right)\left(i-\frac{m}{2}\right) + \sum_{i=\lfloor \frac{m}{2} \rfloor + 1}^{m} \left(l-\frac{1}{l}\right)\left(i-\frac{m}{2}\right) + \frac{5m}{2}$$
$$\leq \frac{m^2+1}{4}\left(l-\frac{1}{l}\right) + \frac{5m}{2} -$$

ℓ_{old}과 ℓ_{new} 값의 비교를 위해 비를 계산하면 다음과 같다.

$$ratio_l = \frac{l_{old}}{l_{\neq w}} \geq \frac{\frac{m(m+1)}{2}\left(l-\frac{1}{l}\right)}{\frac{m^2+1}{4}\left(l-\frac{1}{l}\right)+\frac{5m}{2}}$$

m이 충분히 커질 때에 ratio$_\ell$의 값은 m의 최고 차항의 계수비인 2에 가까워진다. 이로부터 기존의 스케일드 앤 워프트 매칭 거리에서의 엔벨로프를 이루는 최대, 최소 값을 구하기 위해 보는 범위가 중심을 일치시킨 것의 엔벨로프의 그것에 비해 두배 정도라고 할 수 있고 따라서 기존의 것에 비해 새롭게 제시한 엔벨로프가 더 타이트할 것으로 기대할 수 있다. 실제로 실험을 통해 가지치기 성능이 향상됨을 보였다.

Q와 C 사이의 $US_C(Q, C, \ell)$의 하한을 계산하는 하계 함수는 정의 3.3의 하계 함수와 같다. 가지치기 기법을 사용하여 효율적으로 검색하는 경우에도 착오 기각(false dismissal)이 발생하지 않음을 보이기 위하여 새로운 균일 스케일링 거리의 하계 함수가 항상 새로운 균일 스케일링 거리의 하한을 계산함 즉, $LB(Q,C) \leq US_C(Q, C, \ell)$임을 증명한다.

정리 5.1 임의의 두 시퀀스 $Q=[q_1, \cdots, q_m]$과 $C=[c_1, \cdots, c_n]$, 최대 스케일링 상수 ℓ ($\ell \geq 1, n \geq m\ell$)이 주어졌을 때 $LB(Q, C)$는 $US_C(Q, C, \ell)$의 하한을 계산한다.

증명. 새로운 균일 스케일링 거리를 계산하기 위하여 거리를 최소로 하는 Q 전체와 C의 서브 시퀀스 간의 매칭 경로 p를 $p=w_1, \cdots, w_m=(i, j)_1, \cdots, (i, j)_m$이라 하자. 매칭 경로 p의 k번째 요소 $w_k=(i, j)_k=(i_k, j_k)$는 Q의 i_k번째 요소와 C의 j_k번째 요소가 서로 매칭되는 것을 의미하고 균일 스케일링 거리에서는 Q의 각 요소마다 C의 요소들이 대응되므로 모든 경로의 각 요소들에 대하여 $i_k=k$이며 매칭 경로의 길이는 Q의 길이와 같고 따라서 k의 범위는 $1 \leq k \leq m$이다. 길이 q인 C의 서브 시퀀스를 Q와 매칭시키기 위해 길이 m으로 스케일링했을 때 정의 5.1에 의해 Q의 i번째 요소와 매칭되는 C의 요소의 인덱스 j는 $j=\left|\frac{n-q}{2}\right|+\left|i \cdot \frac{q}{m}\right|$이고, 따라서

$$\left|\frac{n}{2}+q\left(\frac{i}{m}-\frac{1}{2}\right)\right| \leq j \leq \left|\frac{n+3}{2}+q\left(\frac{i}{m}-\frac{1}{2}\right)\right|$$

이다. C의 서브 시퀀스의 길이 q는 주어진 최대 스케일링 상수 ℓ에 따라 $m/\ell \leq q \leq m\ell$ 범위에서 변하고 따라서 새로운 균일 스케일링 거리를 계산했을 때 매칭 경로에서 j의 범위는

$$\begin{cases} \left|\frac{n}{2}+lm\left(\frac{i}{m}-\frac{1}{2}\right)\right| \leq j \leq \left|\frac{n+3}{2}+\frac{m}{l}\left(\frac{i}{m}-\frac{1}{2}\right)\right| & i \leq \frac{m}{2} \\ \left|\frac{n}{2}+\frac{l}{m}\left(\frac{i}{m}-\frac{1}{2}\right)\right| \leq j \leq \left|\frac{n+3}{2}+ml\left(\frac{i}{m}-\frac{1}{2}\right)\right| & i > \frac{m}{2} \end{cases}$$

이다. 엔벨로프는 정의 5.2와 같이 정의되므로

$uc_i \geq c_j \quad \therefore q_i - uc_i \leq q_i - c_j$ 이다.

$q_i > uc_i$인 경우 $q_i - uc_i > 0$ 이므로 $(q_i - uc_i)^2 \leq (q_i - c_j)^2$

이고, $q_i < lc_i$ 인 경우에도 마찬가지로 $(q_i - lc_i)^2 \leq (q_i - c_j)^2$

이다. 각각의 i에 대해

$$\begin{cases} (q_i - uc_i)^2 \leq (q_i - c_j)^2 \text{ if } q_i > uc_i \\ (q_i - lc_i)^2 \leq (q_i - c_j)^2 \text{ if } q_i < lc_i \\ 0 \leq (q_i - c_j)^2 \quad otherwise \end{cases}$$

모든 i에 대하여 더하여 $US_C(C, Q, \ell)$과 $LB(Q,C)$를 구하면 $LB(Q,C) \leq US_C(C,Q,\ell)$ 이다.

5.2 중심을 일치시킨 스케일드 앤 워프트 매칭

본 논문에서는 허밍 질의와 데이타베이스의 음악을 비교하기 위하여 새롭게 정의한 중심을 일치시킨 균일 스케일링 거리와 타임 워핑 거리를 결합한 변형된 형태의 스케일드 앤 워프트 매칭을 제시한다. 이 경우 기존의 스케일드 앤 워프트 매칭을 사용한 것과 검색 결과는 같지만 기존의 것에 비해 더욱 타이트한 엔벨로프를 사용하기 때문에 하한이 더욱 타이트하고 따라서 효율적인 검색이 가능하다. 중심을 일치 시킨 스케일드 앤 워프트 매칭 거리는 다음과 같이 정의한다.

정의 5.3 [중심을 일치시킨 스케일드 앤 워프트 매칭 거리] 질의 시퀀스 $Q=[q_1, \cdots, q_m]$, 데이타베이스의 시퀀스 $C=[c_1, \cdots, c_n]$($m \leq n$), 최대 스케일링 상수 ℓ ($\ell \geq 1$, $n \geq \ell m$), 타임 워핑 제한 상수 r이 주어졌을 때, $\lceil m/\ell \rceil \leq q \leq \lceil m\ell \rceil$에 대하여 $C'(m,q)$는 정의 5.1에서처럼 길이가 q이며 C와 중심을 일치시킨 C의 서브 시퀀스를 길이 m이 되도록 스케일링한 것이다. 즉, $C'(m,q)$의 $i(1 \leq i \leq m)$번째 요소 $C'(m,q)_i$는 다음과 같다.

$$C'(m,q)_i = c_{\lfloor (n-q)/2 \rfloor + \lfloor iq/m \rfloor}$$

C와 Q 사이의 중심을 일치시킨 스케일드 앤 워프트 매칭 거리 $SWM_C(C,Q,l,r)$은 다음과 같다. 이 때, cDTW는 제한된 타임 워핑 거리 함수이다.

$$SWM_C(C,Q,l,r) = \min_{\lfloor m/l \rfloor \leq q \leq \lfloor lm \rfloor} cDTW(C'(m,q), Q, r)$$

3장에서 스케일링과 타임 워핑을 함께 적용할 때 타임 워핑 제한 상수에 대해 언급했던 대로 스케일링을 적용하지 않은 타임 워핑 제한 상수를 r′으로 사용하겠다. 중심을 일치시킨 스케일드 앤 워프트 매칭 거리의 하한을 계산하기 위한 엔벨로프와 하계 함수는 다음과 같다.

정의 5.4 [중심을 일치시킨 스케일드 앤 워프트 매칭에서의 엔벨로프] 질의 시퀀스 $Q=[q_1, \cdots, q_m]$, 데이타베이스의 시퀀스 $C=[c_1, \cdots, c_n](m \le n)$, 최대 스케일링 상수 $\ell\,(\ell \ge 1,\ n \ge \ell m)$, 타임 워핑 제한 상수 r이 주어질 때, C의 엔벨로프는 두 시퀀스 $U=[u_1, \cdots, u_m]$와 $L=[l_1, \cdots, l_m]$으로 이루어지며, $1 \le i \le m$인 i에 대하여 각 요소는 다음과 같다.

$$u_i = max(c_{from}, \ldots, c_{to}), \qquad l_i = min(c_{from}, \ldots, c_{to})$$

$$from = \begin{cases} max\left(\left\lfloor \frac{n}{2} + ml\left(\frac{i}{m} - \frac{1}{2}\right)\right\rfloor - r', 1\right) & \text{if } i \le \frac{m}{2} \\ max\left(\left\lfloor \frac{n}{2} + \frac{m}{l}\left(\frac{i}{m} - \frac{1}{2}\right)\right\rfloor - r', 1\right) & \text{if } i > \frac{m}{2} \end{cases}$$

$$to = \begin{cases} min\left(\left\lceil \frac{n+3}{2} + \frac{m}{l}\left(\frac{i}{m} - \frac{1}{2}\right)\right\rceil + r', n\right) & \text{if } i \le \frac{m}{2} \\ min\left(\left\lceil \frac{n+3}{2} + ml\left(\frac{i}{m} - \frac{1}{2}\right)\right\rceil + r', n\right) & \text{if } i > \frac{m}{2} \end{cases}$$

스케일드 앤 워프트 매칭의 하계 함수가 스케일드 앤 워프트 매칭 거리의 하한을 계산하므로 즉, $LB(Q,C) \le SWM_C(Q, C, \ell)$이므로 가지치기 기법을 사용하여 효율적으로 검색하는 경우에도 착오 기각(false dismissal)이 발생하지 않는다.

정리 5.2 두 시퀀스 $Q=[q_1, \cdots, q_m]$과 $C=[c_1, \cdots, c_n]$, 최대 스케일링 상수 $\ell\,(\ell \ge 1, n \ge m\ell)$, 타임 워핑 제한 상수 r′이 주어질 때 $LB(Q, C)$는 $SWM_C(Q, C, \ell, r)$의 하한을 계산한다.

증명. 정리 5.1의 증명과 같은 방법으로 가능하다.

6. 실험 결과

이 장에서는 기존의 스케일드 앤 워프트 매칭과 중심을 일치시킨 스케일드 앤 워프트 매칭을 사용한 유사 검색의 성능을 비교하고 실제 유사 음악 검색 결과를 보인다.

6.1 실험 환경

실험을 위한 프로그램은 C++로 구현하였으며 모든 실험은 팬티엄 4 2.8GHZ CPU와 메인 메모리 512MB가 장착된 컴퓨터에서 Linux 운영체제를 구동시켜 수행하였다.

데이타는 시계열 데이타와 미디 데이타를 사용하였다. 시계열 데이타는 주식 가격 데이타로 야후[16]에서 제공하는 것을 사용하였다. 1985년부터 2006년까지의 126종목을 다운로드 하여 각각을 200차원씩 잘라 2177개의 시계열 데이타를 얻었다. 미디 데이타는 [17-19]에서 팝과 영화 사운드 트랙 위주로 169곡을 모아서 3000차원 시계열 데이타로 변환하였다.

실험은 기존 스케일드 앤 워프트 매칭(SWM)과 중심을 일치시킨 스케일드 앤 워프트 매칭(center-aligned SWM)의 가지치기 성능을 비교하였다. 검색 과정에서 각각에 대해 실제 데이타에서 거리를 계산하는 횟수를 센다. 이는 가지치기 되지 않아 실제 거리를 계산해야 하는 후보 시퀀스의 개수이므로 작을수록 가지치기 성능이 좋다.

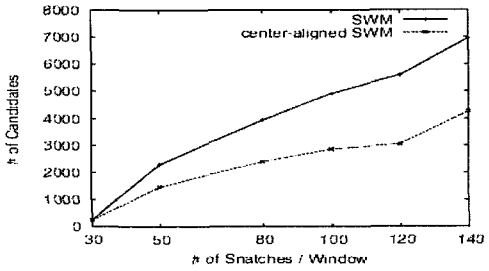
실험에서 변화시킬 수 있는 파라미터는 질의 시퀀스의 길이와 슬라이딩 윈도우의 길이이다. 슬라이딩 윈도우의 길이는 한 마디의 길이와 윈도우를 이루는 마디의 개수의 곱으로 나타내어질 수 있다. 이 때 슬라이딩 윈도우는 한 마디씩 옮겨 가며, 최대 스케일링 상수는 (슬라이딩 윈도우의 길이/질의 시퀀스의 길이)로 주었다. 따라서 질의 시퀀스의 길이가 커지거나 작아지는 것은 슬라이딩 윈도우의 길이가 작아지거나 커지는 것과 같은 효과 이므로 질의 시퀀스의 길이는 고정시키고 한 마디의 길이와 슬라이딩 윈도우를 이루는 마디의 개수를 변화시켜가면서 가지치기 성능을 살펴보도록 하겠다.

6.2 슬라이딩 윈도우를 이루는 마디 개수의 변화에 따른 가지치기 성능의 비교

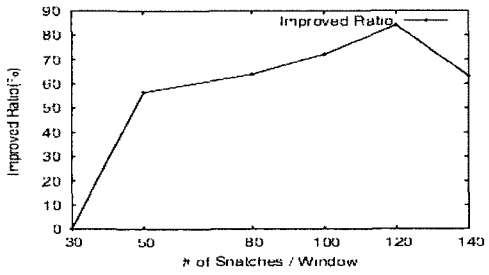
슬라이딩 윈도우를 이루는 마디 개수를 10개부터 140개 까지 변화시켜가면서 가지치기 효과를 비교한 결과를 그림 4,5에 보였다. 마디 길이는 각각 1, 5로, 질의 차수는 30으로 고정하였다. 각 그림의 위쪽 그래프는 유사 검색 수행 시 실제 거리를 계산하게 되는 후보 시퀀스의 개수를 비교한 그래프이고 아래쪽은 기존의 균일 스케일링에 비해 중심을 일치시킨 균일 스케일링에서 후보 시퀀스의 개수가 감소한 정도를 비율로 나타낸 그래프이다. 질의 시퀀스의 길이와 데이타 시퀀스의 길이가 같아서 최대 스케일링 상수가 1이고 마디 길이가 1일 때에는 스케일링이 허용되지 않기 때문에 두 가지의 가지치기 성능이 같음을 그림 4(a), 5(a)에서 확인 할 수 있다. 5장의 $ratio_\ell$식에서 ℓ의 값이 커질수록 기존의 균일 스케일링에 비해 개선된 균일 스케일링의 성능의 좋을 것으로 예상되며 그림 4(b), 5(b)에서 확인 할 수 있다. 4(b)에서 y축은 기존의 균일 스케일링에 비해 중심을 일치시킨 균일 스케일링에서 후보 시퀀스의 개수가 감소한 정도를 '비율'로 나타내고 있다. 따라서 기존의 것보다 성능이 증가하기는 하지만 성능 증가 비율로 보면 감소할 수 있다.

6.3 마디 길이의 변화에 따른 가지치기 성능의 비교

질의 시퀀스의 길이와 슬라이딩 윈도우를 이루는 마디 개수를 고정하고 마디 길이를 변화시켜가면서 비교한 결과를 그림 6,7에 보였다. 주식 데이타는 하나의 데이타의 차원이 200이기 때문에 마디 길이를 1~10까지,

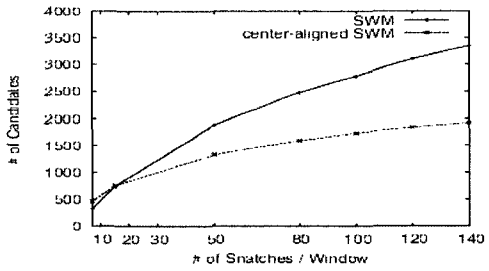


(a) 후보 시퀀스 개수

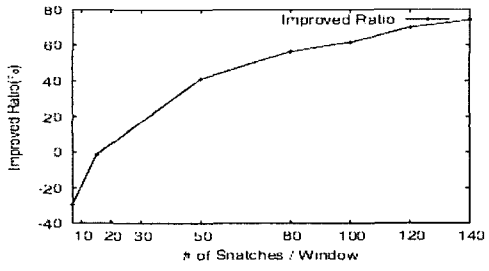


(b) 가지 치기 성능 향상 비율

그림 4 주식 데이타에서 질의 차수 30, 마디 길이 1일 때



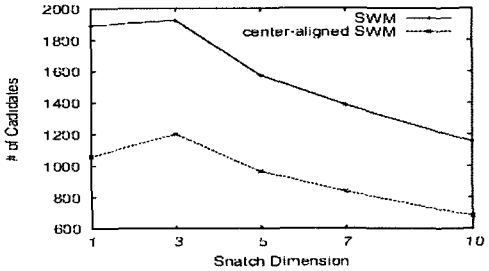
(a) 후보 시퀀스 개수



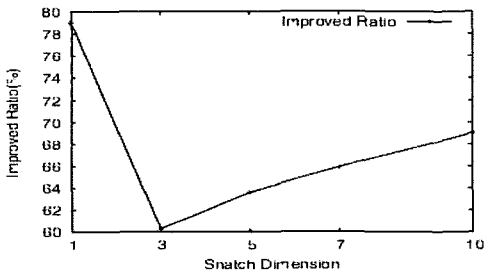
(b) 가지 치기 성능 향상 비율

그림 5 미디 데이타에서 질의 차수 30, 마디 길이 5일 때

미디 데이타는 3000 차원이기 때문에 좀더 길게 1~20까지 변화 시키면서 살펴보았다. 슬라이딩 윈도우 하나에 들어가는 마디의 개수를 n_s라고 표현한다. 질의 시퀀스의 길이는 10, n_s는 20으로 고정시켰다.

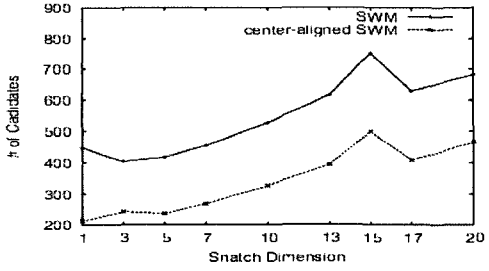


(a) 후보 시퀀스 개수

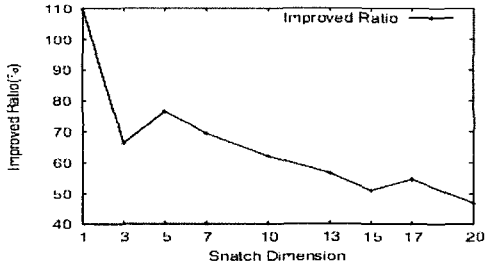


(b) 가지 치기 성능 향상 비율

그림 6 주식 데이타에서 질의 차수가 10, ns는 20일 때



(a) 후보 시퀀스 개수



(b) 가지 치기 성능 향상 비율

그림 7 미디 데이타에서 질의 차수가 10, ns는 20일 때

그림 6의 그래프들을 보면 마디의 차원이 커질수록 후보 시퀀스의 개수가 줄어드는데, 이는 슬라이딩 윈도우가 마디 단위로 옮겨 가는데 마디의 크기가 커지면서 슬라이딩 윈도우의 개수 자체가 줄어들기 때문이다. 중

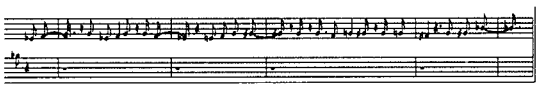
심을 일치시킨 균일 스케일링을 사용했을 때 질의 시퀀스의 차원이 일정하므로 마디의 차원이 커지면 슬라이딩 윈도우의 차원이 커지고 따라서 스케일링 상수 ℓ의 값이 커져서 가지치기 효과가 개선되는 비율이 더욱 높아질 것이다. 그러나 그림 7에서 보이는 것과 같이 마디 단위로 움직이는 슬라이딩 윈도우 때문에 데이타의 서브 시퀀스와 데이타가 중심을 맞추지 못하게 되는 문제가 발생하기 때문에 항상 좋아 지는 것은 아니고 때때로 개선되는 정도는 감소하기도 한다.

6.4 실제 유사 음악 검색 결과

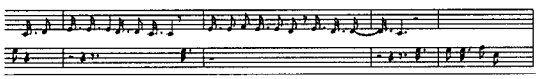
마지막으로 Humming BIRD를 사용한 유사 음악 검색의 결과 예를 보이겠다. 표 1은 BeeGees의 "How Deep Is Your Love"란 곡의 도입부 5마디를 허밍으로 질의하였을 때에 얻은 유사 검색 결과이다. 오프셋은 노래의 몇 번째 마디부터 인지를 나타내는 것이다. 첫 번째와 네 번째 결과는 사용자가 허밍한 "How Deep Is Your Love"의 일부분이 노래 내에서 반복되어 서도 다른 오프셋을 가지는 두 부분이 유사 검색의 결과로 나타난 것이다. 허밍 질의한 부분과 두 번째, 세 번째 결과를 비교해 보기 위해 그림 8에 각 부분의 악보를 보였다. 4장에서 언급했듯이 미디 데이타와 허밍 질의는 정규화 과정을 거친다. 따라서 그림 8의 각 부분은 악보상에서 절대적인 음은 서로 다르지만 음이 변화하는 형태가 거의 비슷하기 때문에 유사한 검색의 결과로 얻어지게 되었다.

표 1 유사 검색 결과

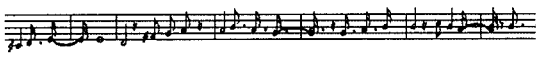
ID	TITLE	OFFSET	DISTANCE
2056	HowDeepIsYourLove	6	5.24851
1022	LetItBe	6	6.17921
6535	MyWay	34	6.22629
2012	HowDeepIsYourLove	55	6.29763
6563	MyWay	62	6.34356



(a) 허밍 질의



(b) Let It Be의 6번째 마디



(c) My Way의 34번째 마디

그림 8 유사 검색 결과로 나온 실제 미디 데이타

7. 결 론

본 논문에서는 허밍 질의와 미디 데이타를 시계열 데이타로 보고 시계열 데이타의 유사성을 검색하는 유사 음악 검색 시스템, Humming BIRD를 제안하고 구현하였다. 음악의 임의의 부분에 대한 허밍 질의를 처리하여 원하는 유사 검색 결과를 얻기 위해 데이타를 슬라이딩 윈도우를 사용하여 인덱싱 하고 허밍 질의 처리에 적절하며 효율적인 검색이 가능한 중심을 일치시킨 스케일드 앤 워프트 매칭을 사용하였다.

실제 데이타에 대해 유사 검색 질의를 수행하여 기존의 스케일드 앤 워프트 매칭에 비해 가지치기 효과가 개선됨을 보였다.

참 고 문 헌

[1] NYU Query by Humming. http://querybyhum.cs.nyu.edu/index.php?p=others/
[2] MIR.http://mirsystems.info/index.php?id=mirsystems
[3] United States Patent 6678680 http://www.freepatentsonline.com/6678680.html
[4] 학교법인영남학원. 허밍과 음성인식을 이용한 음악정보검색방법. 국내 특허 출원 번호 10-2003-0087153.
[5] Alexandra L. Uitdenbogerd and Justin Zobel, "Melodic matching techniques for large music databases," In Proceeding of ACM Multimedia, 57-66, 1999.
[6] Naoko Kosugi, Yuichi Nishihara, Tetsuo Sakata, Masashi Yamamuro and Kazuhiko Kushima, "A practical query-by-humming system for a large music database," In Proceeding of ACM Multimedia, 333-342, 2000.
[7] Asif Ghias, Jonathan Logan, David Chamberlin and Brian C. Smith, "Query by Humming: Musical Information Retrieval in an Audio Database," In Proceeding of ACM Multimedia, 231-236, 1995.
[8] Steffen Pauws, "CubyHum: a fully operational query by humming system," In Proceeding of ISMIR, 2002.
[9] L.Prechelt and R.Typke. An interface for melody input. ACM Transactions on Computer-Human Interaction(TOCHI), 8(2), 133-149, 2001.
[10] Shyamala Doraisamy and Stefan M. Ruger, "Robust Polyphonic Music Retrieval with N-grams," In Journal of Intelligent Information Systems(JIIS), 21(1), 53-70, 2003.
[11] Roger B. Dannenberg and Ning Hu, Understanding Search Performance in Query-by-Humming Systems. In proceeding of ISMIR, 2004.
[12] D. Mazzoni and R. B. Dannenberg. Melody matching directly from audio. In 2nd Annual International Symposium on Music Information Retrieval, 2001.
[13] Y. Zhu and D. Shasha. Warping Indexes with Envelope Transforms for Query by Humming. In

Proc. of ACM SIGMOD Conference, 181-192, 2003.
[14] A.W.Fu, E.J.Keogh, L.Y.H.Lau and C. Ratanamahatana. Scaling and Time Warping in Time Series Querying. In Proc. of the VLDB Conference, 649-660, 2005.
[15] E.J.Keogh, T.Palpanas, V.B.Zordan, D. Gunopulos and M.Cardle. Indexing Large Human-Motion Databases. In Proc. of VLDB Conference, 780-791, 2004.
[16] Yahoo Finance. http://finance.yahoo.com/q/hp?s=GE
[17] Midi Database. http://www.mididb.com/
[18] Midi4U. http://www.midi4u.com/
[19] Free Midi Zone. http://www.free-midi.org/

이 혜 환

2004년 서울대학교 전기컴퓨터공학부 졸업(학사). 2006년 서울대학교 전기컴퓨터공학부 졸업(석사). 관심분야는 데이타마이닝, 데이타베이스

심 규 석

정보과학회논문지 : 데이타베이스
제 34 권 제 4 호 참조

박 형 민

2003년 서울대학교 전기컴퓨터공학부 졸업(학사). 2004년~현재 서울대학교 전기컴퓨터공학부 박사과정. 관심분야는 데이타마이닝, 데이타베이스