향상된 균일 스케일링을 이용한 유사 음악 검색시스템

이혜환° 심규석
서울 대학교 전기 컴퓨터 공학부
hhlee @kdd.snu.ac.kr°, shim@ee.snu.ac.kr

A Similar Music Retrieval System using Improved Uniform Scaling

Hyehwan Lee° Kyuseok Shim
School of Electrical Engineering and Computer Sciences, Seoul National University

요 약

허밍을 통한 유사 검색 질의가 주어질 때 효과적으로 음악 데이터베이스를 검색하는 시스템에 대한 연구는 다양한 방향으로 진행되어 왔다. 최근에는 음악 데이터와 허밍 질의를 시계열 데이터로 보고 시계열 데이터 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 고려한 균일 스케일링(Uniform Scaling)을 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 효율적인 검색을 위해 중심을 일치시킨(center-aligned) 균일 스케일링을 제안하고 이 거리의 하한을 계산하는 하계 함수를 사용하여 탐색 공간(search space)을 효과적으로 줄여 더 빠르고 효과적인 유사 검색을 가능하도록 하였으며 실험을 통해 중심을 일치시킨된 균일 스케일링이 이전과 같은 검색 결과를 얻으면서도 효과적으로 검색함을 탐색 공간을 줄이는 가지치기 성능을 비교함으로써 보였다.

1. 서 론

기존의 널리 사용되어 오던 유사 음악 검색 시스템들은 대부분 키워드 기반의 검색 시스템이어서 사용자가 제목이나 가수, 가사와 같은 음악에 관한 텍스트 형태의 메타 정보를 갖지 못한 경우에 멜로디로만 어렴풋이 기억하고 있는 음악을 검색할 수가 없었다. 실제로 누구나 한번쯤은 머리 속에 맴도는 멜로디만으로 노래를 검색할 수 없어 답답했던 경험이 있을 것이다. 이를 해결하기 위하여 연구되고 있는 것이 내용에 기반한 질의 시스템(Query by content)인 허밍 질의를 통한 유사 음악 검색 시스템이다. 사용자는 마이크를 통해 자신이 기억하는 노래의 일부를 허밍하고 시스템은 음악 데이터베이스를 검색하여 허밍 질의와 비슷한 부분을 포함하는 음악의 목록을 결과로 보여준다.

휴대폰에서 사용할 수 있는 멀티미디어 컨텐츠와 서비스가 다양해지면서 통신 업계에서는 휴대폰을 통해 허밍 입력을 받아 원하는 음악을 검색하고 듣는 서비스를 개발하여 이를 제공하기 시작했다. 국내 최대 이동 통신사인 SK텔레콤에서는 디지털 콘텐츠 검색 기술 전문 업체인 나요 미디어와 함께 전화를 통해 노래의 일부분을 허밍하면 이와 유사한 원하는 음악을 찾아주는 허밍 기반 음악 검색 엔진을 개발, 세계 최초로 상용화하여 서비스를 시작하였다[1]. 일본의 NTT 도코모, 도시바 등에서도 관련 기술이 개발되면서[2,3] 허밍 질의를 통한 유사 음악 검색 시스템은 본격적으로 모바일 음악 서비스에 적용될 예정이다. 이러한 시스템은 휴대폰 이외에도 음반 매장이나 노래방과 같은 곳에서 제목이나 가수의 정보를 기억하지 못해 음악을 찾지 못했던 사용자들에게 유용하게 사용될 것이며 작곡가들에게는 자신이 작곡한 멜로디의 창작성 여부를 검증해 보는데 사용될 수도 있다.

효과적인 허밍 질의를 통한 유사 음악 검색 시스템에 대한 관심이 높아지면서 국내외 대학 및 연구소에서 이와 관련된 다양한 프로젝트를 진행하여 [4,5]등의 데모 버전을 제공하고 있으며, [6,7]과 같은 관련 특허들도 출원되고 있다. 기술의 상용화가 진행되면서 언론에서도 그 연구 결과의 유용성에 주목하고 있다[1,8]. 그러나 현재 시스템들은 대용량 음악 데이터에서 다양한 허밍 질의에 대해 원하는 결과를 돌려주지 못하거나 검색 시간이 오래 걸리는 등 아직 많은 문제점을 가지고 있다.

본 논문에서는 허밍 질의와 미디 데이터를 시계열 데이터로 보아 시계열 데이터의 유사성 검색 문제를 해결하는 입장에서 허밍 질의를 통한 유사 음악 검색 시스템인 Humming BIRD를 구현하였다. 임의의 부분에 대한 허밍 질의를 처리하기 위해 미디 데이터를 슬라이딩 윈도우를 사용하여 인덱싱하였으며 허밍 질의에 적합한 거리 함수를 선택하고 개선하여 효율적으로 검색 할 수 있는 방법을 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 허밍 질의의 통한 유사 음악 검색 시스템들을 간단히 소개하고 장단점을 살피고 3장에서는 본 논문이 바탕을 두고 있는 기존의 거리 척도인 균일 스케일링(Uniform Scaling)과 효율적인 유사 검색에 사용되는 이 거리 척도의 하계 함수(lower bounding function)를 설명한다. 4장에서는 전체적인 허밍 버드 시스템을 설명하고 5장에서는 본 논문에서 새롭게 제시하는 중심을 일치시킨 균일 스케일링과 하계 함수를 정의하고 설명한다. 6장에서는 기존의 균일 스케일링을 사용한 것과 성능을 비교하는 실험을 보인다.

2. 관 련 연 구

허밍 질의를 통한 유사 음악 검색 시스템은 멀티미디어 데이터베이스 분야에서 다양한 방향으로 연구되어 오고 있다. 이 장에서는 기존에 제시되었던 중요한 몇 가지 접근 방법들을 소개한다.

2.1 음조의 유사성에 기반한 관련 연구

음조란 음의 상대적인 높이의 변화를 뜻하는 것으로서 멜로디

· 본 연구는 2006년도 정보통신부 지원 대학 IT 연구센터 육성지원사업의 연구비 지원으로 수행하였습니다.

를 표현하는데 사용되는 한 가지 방법이다. 음조는 보통 몇 개의 정해진 알파벳으로 나타내어지는데[9], 예를 들어 많이 사용되는 알파벳 중 한 종류가 'U', 'D', 'S' 이다. 음표의 음이 이전 음표에 비해 높은 경우에는 'U(up)', 낮은 경우에는 'D(down)', 비슷한 경우에는 'S(same)'와 같이 표현한다. 데이터베이스의 음악도 마찬가지 방법으로 음조를 표현한다. 이렇게 멜로디는 몇 개의 알파벳으로 이루어진 문자열로 표현하고, 편집 거리(Edit distace)를 사용하여 유사한 멜로디로 이루어진 음악을 검색할 수 있다. 이 방법은 사용자가 허밍을 부정확하게 하는 경우에도 멜로디를 잘 추출할 수 있지만 이전 음표에 대비한 상대적인 높낮이만으로 멜로디를 표현하기 때문에 실제 멜로디를 표현하기에는 정보가 부족하여 정확한 유사 검색 결과를 얻기가 힘들다. 또한 허밍 질의로부터 분리된 음표를 얻는 신뢰할만한(reliable) 알고리즘이 아직 알려진 것이 없어 질의에 따라 어떤 알고리즘을 적용하느냐 하는 것이 유사성 검색 결과를 좌우하는 경우가 많은 문제가 있다.

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[10]. 허밍 질의에 피치 추출 알고리즘을 적용하여 연속적인 피치의 시퀀스를 얻고 미디 파일의 멜로디 정보로부터 역시 피치의 시퀀스를 얻었다. 여기에 타임 워핑 거리를 사용하여 유사 검색을 수행한 결과 이전의 음조에 기반한 시스템에 비해 훨씬 좋은 결과를 얻었다. 그러나 긴 시퀀스에 대해 타임 워핑 거리를 계산하기 위한 동적 프로그래밍(dynamic programming)을 수행하는데 시간이 매우 오래 걸려서 실제로 허밍 시스템에서 사용하기 힘들다.

2.2 시계열 데이터 유사 검색의 기법을 적용한 관련 연구

시계열 데이터베이스에서의 유사 검색은 데이터 마이닝 분야에서 수년 동안 활발히 연구되어 왔던 분야이다. 다양한 분야의 데이터에서 의미 있는 유사 검색을 위한 거리 함수들이 제시되었고 효과적인 검색을 가능하게 하는 다차원 인덱스 구조를 사용하기 위한 차원 감소(dimensionality reduction) 기법들이 연구되었다[11,12,13]. 따라서 허밍 질의와 음악 데이터를 시계열 데이터로 보고 기존의 기법들을 적용시키려는 연구들이 진행되어 왔다. [14]에서는 시계열 데이터베이스에서 사용되는 유사 검색 기법을 사용하여 허밍 질의 처리 시스템을 구현하였다. 이 논문에서는 유사성을 측정하는 거리 척도로 타임 워핑 거리(Dynamic Time Warping)를 사용하였다. 효과적으로 검색하기 위해 [15]에서 제시한 타임 워핑 거리에 대한 하계 함수를 개선하여 적용하였으며 차원 감소 기법을 사용하여 인덱싱을 할 때 여러 차원 감소 기법을 일반적으로 적용할 수 있음을 증명하였다.

[16, 17]에서는 허밍질의 시스템에 확장, 적용 시킬 수 있는 거리 척도인 균일 스케일링(Uniform Scaling) 거리, 스케일드 앤 워프트 매칭(Scaled and Warped Matching) 거리와 각각의 하한을 계산하는 하계 거리 함수를 정의하고 이를 사용하여 효과적으로 유사 검색을 수행할 수 있는 기법을 제시하였다.

3. 기 본 개 념

이 장에서는 시계열 데이터 유사 검색에 사용되어 왔던 기존의 거리 척도 중 본 논문의 유사 음악 검색 시스템에서 사용하는 거리 척도인 균일 스케일링 거리의 정의와 이를 사용하여 효율적으로 유사 검색을 수행하기 위한 하계 함수를 살펴본다. [17]에서 제안된 균일 스케일링 거리는 두 시퀀스의 거리를 계산할 때 전체적인 시간 축 상의 확대와 축소를 고려하는 거리 척도로 허밍 질의와 같이 질의 시퀀스가 유사한 시퀀스에 비해 전체적으로 빠르게 혹은 느리게 생성될 수 있을 때에 유용하게 사용할 수 있다. 균일 스케일링 거리의 정의는 다음과 같다.

정의 3.1[균일 스케일링 거리]

질의 시퀀스 Q= [q_1, q_2, ... , q_m], 데이터베이스의 시퀀스 C= [c_1, c_2, ..., .c_n](m≤ n), 최대 스케일링 상수 $\ell(\ell \geq 1,\ n \geq \ell m)$ 이 주어졌을 때, $\lceil m/\ell \rceil \leq q \leq \lceil m\ell \rceil$ 인 q에 대하여 C(m, q)는 길이 q인 C의 접두 서브 시퀀스 [c_1, c_2, ... , c_q]가 질의 시퀀스 Q의 길이 m과 같아지도록 스케일링된 것이다. 즉, $C(m,q)$의 i(1≤i≤m)번째 요소 $C(m,q)_i$는 다음과 같이 표현된다.

$C(m,q)_i = c_{\lceil iq/m \rceil}$

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

$US(C,Q,l) = \min_{\lceil m/\ell \rceil \leq q \leq \lceil l m \rceil} D(C(m,q), Q)$

가능한 각각의 q에 대하여 유클리디언 거리를 구하는데 O(m)의 시간이 걸리므로 이 거리 값을 구하는 데 걸리는 시간 복잡도는 $O(m^2)$이다. 모든 데이터에 대해 질의 시퀀스와의 균일 스케일링 거리를 계산하는 것은 상당히 큰 CPU 비용을 필요로 한다. 따라서 빠른 시간 내에 계산할 수 있으며 실제 거리의 하한을 계산하는 하계 함수를 정의하고 이 하계 함수를 사용하여 가지치기 하는 기법을 보편적으로 사용한다.

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

정의 3.2 [균일 스케일링에서의 엔벨로프]

질의 시퀀스 Q= [q_1, q_2, ... , q_m], 데이터베이스의 시퀀스 C= [c_1, c_2, ..., .c_n](m≤ n), 최대 스케일링 상수 $\ell(\ell \geq 1,\ n \geq \ell m)$ 이 주어졌을 때, 데이터 시퀀스 C의 엔벨로프는 두 시퀀스 UC=[uc_1, uc_2, ... , uc_m]와 LC=[lc_1, lc_2, ... , lc_m]으로 이루어지며, 1≤i≤m인 i에 대하여 각 요소는 다음과 같다.

$uc_i = max(c_{\lceil i/\ell \rceil}, ..., c_{\lceil i\ell \rceil})$, $lc_i = min(c_{\lceil i/\ell \rceil}, ..., c_{\lceil i\ell \rceil})$

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

정의 3.3 [하계 함수]

두 시퀀스 Q= [q_1, q_2, ... , q_m], C= [c_1, c_2, ..., .c_n]와 C의 엔벨로프 U=[u_1, u_2, ... , u_m]와 L=[l_1, l_2, ... , l_m]이 주어졌을 때, 하계 함수 $LB(Q,C)$는 다음과 같이 정의된다.

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

4. 전체 시스템의 개요

이 장에서는 본 논문이 Humming Bird를 통하여 해결 하고자 하는 문제 정의 및 전체적인 시스템 구성과 각 요소에 대한 간략한 설명을 하겠다.

4.1 문제 정의

본 논문의 허밍을 통한 유사 음악 검색 시스템이 해결하고자 하는 문제는 음악 데이터베이스와 노래의 일부분을 허밍한 질의가 주어졌을 때, 허밍 질의와 유사한 부분을 갖는 음악을 검색하는 것이다.

4.2 전체적인 시스템 구성

본 논문에서 제안하는 허밍 질의 처리 시스템, Humming BIRD는 미디 데이터 처리 모듈, 허밍 입력 처리 모듈, 유사 검색 수행 모듈들로 구성된다.

미디 데이터 처리 모듈은 미디 데이터를 전처리하여 시계열 데이터를 얻는다. 음악을 일련의 이벤트로 표현하는 미디 파일로부터 멜로디를 추출하여 음의 시퀀스로 변환하고 헤더 정보를 이용하여 한 마디가 일정한 개수의 값으로 나타나도록 샘플링한다. 이 때 쉼표는 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 데이터로 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 데이터를 정규화하며 전체적인 빠르기나 부분적인 빠르기의 변화도 고려하면서도 효율적으로 검색하도록 균일 스케일링을 개선하여 적용하였다. 미디 파일을 처리한 결과로 얻어지는 시퀀스의 슬라이딩 윈도우로부터 차원 감소 기법을 사용하여 피쳐(feature)를 추출하고 이를 사용하여 R*-tree와 같은 다차원 인덱스를 구축한다. 질의 시퀀스로부터도 피쳐를 추출하고, 인덱스 구조를 사용하여 실제 유사한 데이터일 가능성이 있는 데이터만 실제 데이터를 디스크로부터 읽고 실제 비교를 수행한다. 5장에서 좀더 자세히 설명하겠다.

5. 향상된 균일 스케일링을 사용한 유사 검색

이 장에서는 좀 더 효과적으로 가지치기를 하고자 기존의 균일 스케일링 거리를 변형하여 중심을 일치시킨 균일 스케일링 거리를 제안하며 이를 사용하여 효율적으로 유사 검색을 수행하고자 사용하는 하계 함수를 정의하고 실제 거리의 하한을 계산함을 보인다.

실제 거리의 하한을 계산하는 하계 함수를 사용하여 가지치기를 하는 것이 효과적이기 위해서는 하계 함수가 실제 거리의 하한을 타이트하게 계산하여야 한다. 만약 하계 함수가 실제 거리의 하한을 계산한다고 해도 그 하한이 너무 느슨한 경우에는 하계 함수를 사용하여 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 하한을 계산하며 계산도 빠르다. 하지만 모든 데이터와 질의에 대하여 하한이 0 이므로 전혀 가지치기를 하지 못하고 실제 거리를 계산해야 한다. 따라서 하계 함수를 타이트하게 정의 하는 것은 가지치기 성능을 좌우하여 효율적인 유사 검색을 가능하게 한다. 따라서 본 논문에서는 효율적으로 유사 검색을 수행하고자 좀 더 타이트한 하계 함수와 이를 위해 실제 데이터를 더욱 타이트하게 감싸는 엔벨로프를 제안한다.

예 5.1 8차원 질의 시퀀스 $Q=[q_1, q_2, \ldots, q_8]$와 16차원 데이터 시퀀스 $C=[c_1, c_2, \ldots, c_{16}]$가 주어지고 ℓ은 2일 때 기존의 균일 스케일링에서의 엔벨로프는 아래와 같이 정해진다.

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

예 5.1와 같이 기존의 엔벨로프는 각 차원 i에 대해 $c_{\lceil i/\ell \rceil}$부터 $c_{\lceil i\ell \rceil}$ 사이의 최대 값과 최소 값을 uc_i와 lc_i로 정한다. i가 커질수록 uc_i와 lc_i를 정하기 위해서 봐야 하는 범위($c_{\lceil i/\ell \rceil}$부터 $c_{\lceil i\ell \rceil}$)가 길어지므로 i가 커질수록 엔벨로프가 느슨해짐을 볼 수가 있다. 기존의 균일 스케일링이 데이터시퀀스의 접두 서브 시퀀스를 가장 앞부분을 고정시키고 스케일링 하는 형태이기 때문에 뒤쪽으로 갈수록 질의 시퀀스의 하나의 요소와 매칭될 수 있는 데이터시퀀스의 범위가 넓어져서 생기는 문제이다. 따라서 본 논문에서는 이 점에 관심을 갖고 좀 더 타이트한 하계 함수를 얻어 효과적인 가지치기를 하고자 변형된 형태의 균일 스케일링 거리 함수와 이것의 하한을 계산하는 하계 함수를 얻기 위한 좀 더 타이트한 엔벨로프를 제안하고자 한다. 스케일링을 할 때 고정시키는 부분으로부터 멀어질수록 매칭되는 범위가 넓어져 엔벨로프가 느슨해지므로 스케일링을 할 때에 시퀀스의 가운데를 고정시키고 양쪽으로 균일하게 늘리는 방법, 즉 중심을 일치시킨 균일 스케일링(center-aligned Uniform Scaling)을 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. 예를 통해 중심을 일치시킨 서브 시퀀스를 설명한다.

예 5.2 길이 8인 시퀀스 $C=[c_1, c_2, \ldots, c_7, c_8]$ 이 주어졌을 때 길이 $q(1 \le q \le 8)$에 따른 C와 중심을 일치 시킨 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 유클리디언 거리를 구한 것을 중심을 일치시킨 균일 스케일링 거리로 정의한다.

정의 5.1 [중심을 일치시킨 균일 스케일링 거리]
질의 시퀀스 $Q=[q_1, q_2, \ldots, q_m]$, 데이터베이스의 시퀀스 $C=[c_1, c_2, \ldots, 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 길이가

q이며 C와 중심을 일치시킨 C의 서브 시퀀스를 질의 시퀀스의 길이 m과 같아지도록 균일 스케일링 한 것이다. 즉, $C(m,q)$의 i(1≤i≤m)번째 요소 $C(m,q)_i$는 다음과 같이 표현된다.

$$C(m,q)_i = q_{\lceil (n-q)/2 \rceil + \lceil iq/m \rceil}$$

이 때 C, Q 간의 중심을 일치시킨 균일 스케일링 거리 $US_c(C,Q,l)$은 다음과 같다. D는 유클리디언 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 데이터 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 공간(search space)을 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_2, ... , q_m], 데이터베이스의 시퀀스 C= [c_1, c_2, ..., .c_n](m≤ n), 최대 스케일링 상수 l(l≥1, n≥lm) 이 주어졌을 때, 데이터 시퀀스 C의 엔벨로프는 두 시퀀스 UC=[uc_1, uc_2, ... , uc_m]와 LC=[lc_1, lc_2, ... , lc_m]으로 이루어지며, 1≤i≤m인 i에 대하여 각 요소는 다음과 같다.

$uci = max(c_{from}, ..., c_{to})$, $\quad lc_i = min(c_{from}, ..., c_{to})$

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

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

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 양끝으로 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 간단한 예를 통해 예상할 수 있고 실제로 실험을 통해 가지치기 성능이 향상됨을 보였다.

Q와 C 사이의 US_C(Q, C, l)의 하한을 계산하는 하계 함수는 정의 3.3의 하계 함수와 같다. 가지치기 기법을 사용하여 효율적으로 검색하는 경우에도 착오 기각이 발생하지 않음을 보이기 위하여 새로운 균일 스케일링 거리의 하계 함수가 항상 새로운 균일 스케일링 거리의 하한을 계산함 즉, LB(Q,C)≤ US_C(Q, C, l)임을 증명한다.

정리 5.1
임의의 두 시퀀스 Q=[q_1, q_2, ..., q_m]과 C=[c_1, c_2, ... , c_n], 최대 스케일링 상수 l(l≥1,n≥ml)이 주어졌을 때 LB(Q, C)는 US_C(Q, C, l)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, w_2, ..., w_m=(i, j)$_1$, (i, j)$_2$, ..., (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=k이며 매칭 경로의 길이는 Q의 길이와 같고 따라서 k의 범위는 1≤k≤m 이다.

C의 길이 q인 서브 시퀀스를 Q와 매칭시키기 위해 길이 m으로 스케일링 했을 때 정의 5.1에 의해 Q의 i번째 요소와 와 매칭되는 C의 요소의 인덱스 j는 $j = \left\lceil \frac{n-q}{2} \right\rceil + \left\lceil i \cdot \frac{q}{m} \right\rceil$ 이고,

따라서 $\left\lceil \frac{n}{2} + q\left(\frac{i}{m} - \frac{1}{2}\right) \right\rceil \le j \le \left\lfloor \frac{n+3}{2} + q\left(\frac{i}{m} - \frac{1}{2}\right) \right\rfloor$

이다. C의 서브 시퀀스의 길이 q는 주어진 최대 스케일링 상수 l에 따라서 m/l≤q≤ml 범위에서 변하고 따라서 새로운 균일 스케일링 거리를 계산 했을 때 매칭 경로에서 j의 범위는

$$\begin{cases} \left\lceil \frac{n}{2} + lm\left(\frac{i}{m} - \frac{1}{2}\right) \right\rceil \le j \le \left\lfloor \frac{n+3}{2} + \frac{m}{l}\left(\frac{i}{m} - \frac{1}{2}\right) \right\rfloor & i \le \frac{m}{2} \\ \left\lceil \frac{n}{2} + \frac{l}{m}\left(\frac{i}{m} - \frac{1}{2}\right) \right\rceil \le j \le \left\lfloor \frac{n+3}{2} + ml\left(\frac{i}{m} - \frac{1}{2}\right) \right\rfloor & i > \frac{m}{2} \end{cases}$$

이다. 엔벨로프는 정의5.2와 같이 정의되므로

$uc_i \ge c_j \quad \therefore q_i - uc_i \le q_i - c_j$ 이다.

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

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

이다. 각각의 i에 대해

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

모든 i에 대하여 더하여 US_C(C, Q, l)과 LB(Q,C)를 구하면 LB(Q,C) ≤ US_C(C,Q,l) 이다.

6. 실험 결과

이 장에서는 기존의 균일 스케일링과 중심을 일치시킨 균일 스케일링을 사용한 유사 검색의 성능을 비교한다.

6.1 실험 환경

실험을 위한 프로그램은 C++로 구현하였으며 모든 실험은 팬티엄 4 2.8GHZ CPU와 메인 메모리 512MB가 장착된 컴퓨터

에서 Linux 운영체제를 구동시켜 수행하였다.

데이터는 인터넷 웹사이트들[18,19,20]에서 팝과 영화 사운드트랙 위주로 유명한 169곡의 미디 파일을 모았다. 각각의 곡은 3000차원 시계열 데이터로 변환하였다.

실험에서는 기존 균일 스케일링(US)과 중심을 일치시킨 균일 스케일링(center-aligned US)의 가지치기 성능을 비교하였다. 검색 과정에서 각각에 대해 실제 데이터를 읽고 거리를 계산하는 횟수를 센다. 이는 가지치기 되지 않아 실제 거리를 계산해야 하는 후보 시퀀스(candidate sequences)의 개수이므로 작을수록 가지치기 성능이 좋다.

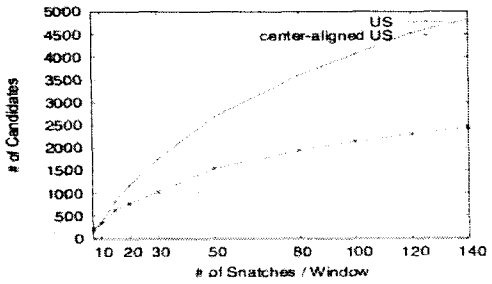
실험에서 변화시킬 수 있는 파라미터는 질의 시퀀스의 길이와 슬라이딩 윈도우의 길이이다. 슬라이딩 윈도우의 길이는 한 마디의 길이와 슬라이딩 윈도우를 이루는 마디 개수의 곱으로 나타내어질 수 있다. 이 때 슬라이딩 윈도우는 한 마디씩 옮겨가며, 최대 스케일링 상수는 (슬라이딩 윈도우의 길이/질의 시퀀스의 길이)로 주었다. 따라서 질의 시퀀스의 길이가 커지거나 작아지는 것은 슬라이딩 윈도우의 길이가 작아지거나 커지는 것과 같은 효과 이므로 질의 시퀀스의 길이는 고정시키고 한 마디의 길이(Snatch dimension)와 슬라이딩 윈도우를 이루는 마디의 개수(number of snatches per sliding window)를 변화 시켜가면서 가지치기 성능을 살펴보도록 하겠다.

6.2 실험 결과

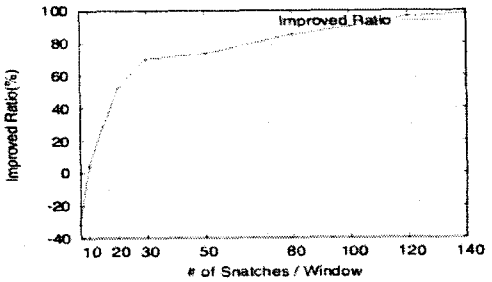
6.2.1 슬라이딩 윈도우를 이루는 마디 개수의 변화에 따른 가지치기 성능의 비교

슬라이딩 윈도우를 이루는 마디 개수를 10개부터 140개 까지 변화시켜가면서 가지치기 효과를 비교한 결과를 그림 1에 보였다. 마디 길이는 각각 5로, 질의 차수는 30으로 고정하였다. 그림 1(a)는 유사 검색 수행 시 실제 거리를 계산하게 되는 후보 시퀀스의 개수를 비교한 그래프이고 그림 1(b)는 기존의 균일 스케일링에 비해 중심을 일치시킨 균일 스케일링에서 후보 시퀀스의 개수가 감소한 정도를 비율로 나타낸 그래프이다.

최대 스케일링 상수의 값이 커질수록 스케일링 될 수 있는 범위가 넓어지므로 엔벨로프가 느슨해지고 따라서 가지치기 효과가 떨어지게 될 것임을 예상할 수 있다. 그리고 실험 결과에서도 질의 차수와 마디의 차수가 일정할 때 마디 개수가 증가함에 따라 후보 시퀀스의 개수가 대체로 증가함을 볼 수 있다. 5장의 $ratio_\ell$식에서 ℓ의 값이 커질수록 기존의 균일 스케일링에 비해 개선된 균일 스케일링의 성능의 좋을 것으로 예상되며 그림 1(b), 2(b)에서 확인 할 수 있다.



(a) 후보 시퀀스 개수

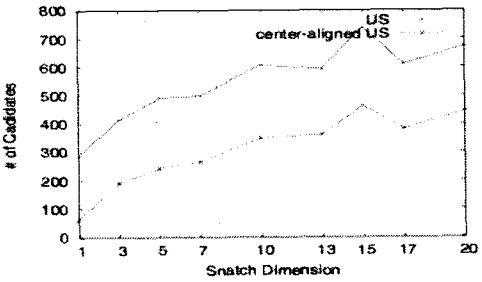


(b) 후보 시퀀스 개수 감소 비율

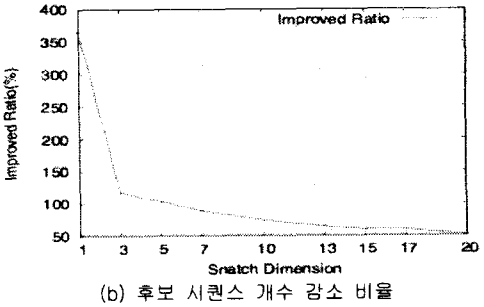
그림1 슬라이딩 윈도우를 이루는 마디 개수의 변화에 따른 가지치기 성능 비교

6.2.2 마디 길이의 변화에 따른 가지치기 성능의 비교

질의 시퀀스의 길이와 슬라이딩 윈도우를 이루는 마디 개수를 고정하고 마디 길이를 1차원부터 20차원까지 변화시켜가면서 비교한 결과를 그림 2에 보였다. 슬라이딩 윈도우 하나에 들어가는 마디의 개수를 n_s라고 표현한다. 질의 시퀀스의 길이는 10, n_s는 20으로 고정시켰다. 그림 2(a)는 유사 검색 수행 시 실제 거리를 계산하게 되는 후보 시퀀스의 개수를 비교한 그래프이고 그림 2(b)는 기존의 균일 스케일링에 비해 중심을 일치시킨 균일 스케일링에서 후보 시퀀스의 개수가 감소한 정도를 비율로 나타낸 그래프이다.



(a) 후보 시퀀스 개수



(b) 후보 시퀀스 개수 감소 비율

그림2 마디 길이 변화에 따른 가지치기 성능 비교

그림 2의 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. 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 동시에 마디 단위로 움직이는 슬라이딩 윈도우 때문에 데이터의 서브 시퀀스와 데이터가 중심을 맞추지 못하게 되는 문제가 발생하기 때문에 항상 좋아 지는 것은 아니고 때때로 개선되는 정도는 감소하기도 한다.

7. 결론 및 향후 연구

본 논문에서는 허밍 질의와 마디 데이터를 시계열 데이터로 보고 시계열 데이터의 유사성 검색을 수행하는 관점에서 유사 음악 검색 시스템, Humming BIRD를 제안하고 구현하였다. 음악의 임의의 부분에 대한 허밍 질의를 처리하여 원하는 유사 검색 결과를 얻기 위해 데이터를 슬라이딩 윈도우를 사용하여 인덱싱 하고 허밍 질의 처리에 적절하며 효율적인 검색이 가능한 중심을 일치시킨 균일 스케일링을 사용하였다.

실제 데이터에 대해 유사 검색 질의를 수행하여 기존의 균일 스케일링에 비해 개선된 균일 스케일링의 가지치기 효과가 개선됨을 보였다. 그러나 마디를 이루는 데이터의 개수가 커지거나 스케일링 상수가 거의 1에 가까운 경우 가지치기 효과가 나빠지는 경우도 발생하는 문제점이 발견되었고 이 문제를 개선하는 것에 관한 연구가 필요하다. 본 논문에서는 임의의 부분에 대한 허밍 질의에 대해서 잘 처리할 수 있도록 하기 위해 슬라이딩 윈도우를 사용하였지만 기존의 연구에서처럼 대부분의 질의는 사용자가 기억하기 쉬운 후렴구와 같이 반복되는 부분이나 음악의 주요 멜로디 부분일 가능성이 높다. 이런 부분을 추출하여 빠르게 우선 검색을 할 수 있도록 인덱스를 구축하는 것에 관한 연구도 진행되고 있는데 이를 시스템에 적용시킬 경우 음악의 주요 부분을 질의하는 많은 경우에 대해 좀 더 빠른 검색이 가능해질 것으로 기대된다. 또한 슬라이딩 윈도우를 사용하여 서브 시퀀스 매칭을 할 때에 비슷한 슬라이딩 윈도우끼리 묶어서 효율적으로 인덱싱하는 것에 관한 알고리즘[19]을 허밍 질의 처리 시스템에도 적용하여 효율성을 높일 수 있을 것으로 생각된다. 실제로 수 만곡의 음악 데이터에서 검색을 빠르게 수행하기 위해서 기존의 제목과 가수, 장르 등의 음악 정보를 사용하여 검색을 하던 시스템과 결합하여 장르나 제목에 들어가는 단어 등을 사용하여 먼저 후보 데이터 범위를 줄인 다음에 유사 검색을 수행하는 방법도 효과적일 것으로 생각된다.

8. 참고 문헌

[1]서울 경제 신문. http:// economy.hankooki.com/lpage/industry/200602/e2006021916083670260.htm
[2]Yahoo! Music News.
http://au.launch.yahoo.com/0410 29/11/1tkf.html
[3]the INQUIRER.
http://www.theinquirer.net/default. aspx?article=15505
[4]NYU Query by Humming.
http://querybyhum.cs.nyu.edu/index.php?p=others/
[5]MIR. http://mirsystems.info/index.php?id=mirsystems
[6]United States Patent 6678680
http://www.freepatentsonline.com/6678680.html
[7]학교법인영남학원. 허밍과 음성인식을 이용한 음악정보검색방법. 국내 특허 출원 번호 10-2003-0087153
[8]MusicDB: A Query by Humming System.
http://web.mit.edu/edmond/www/projects/musicdb.html
[9]L.Prechelt and R.Typke. An interface for melody input. ACM Transactions on Computer-Human Interaction(TOCHI), 8(2),133-149, 2001
[10]D.Mazzoni and R.B.Dannenberg. Melody matching directly from audio. In 2nd Annual International Symposium on Music Information Retrieval, 2001.
[11]R.Agrawal, C.Faloutsos and A.N.Swami. Efficient Similarity Search In Sequence Databases. In Proc. of Foundations of Data Organization and Algorithm(FODO), 69-84, 1993
[12]K.Chan and A.W. Fu. Efficient Time Series Matching by Wavelets. In Proc. of International Conference on Data Engineering(ICDE),126-133, 1999
[13]K.Chakrabarti, E.J.Keogh, S.Mehrotra and M.J. Pazzani. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In ACM Transactions on Database Systems(TODS), 27(2), 188-228, 2002
[14]Y.Zhu and D.Shasha. Warping Indexes with Envelope Transforms for Query by Humming. In Proc. of ACM SIGMOD Conference, 181-192, 2003
[15]E.J.Keogh and C.Ratanamahatana. Exact indexing of dynamic time warping. In Knowledge and Information Systems, 7(3), 358-386, 2004
[16]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
[17]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
[18]Midi Database. http://www.mididb.com/
[19]Midi4U. http://www.midi4u.com/
[20]Free Midi Zone. http://www.free-midi.org/