韓國컴퓨터情報學會 論文誌
第14卷 第12號, 2009. 12.

2009-14-12-2-2

가우시안의 차를 이용하여 검색속도를 향상한 최소 오디오 핑거프린팅

권 진 만*, 고 일 주**, 장 대 식***

Search speed improved minimum audio fingerprinting using the difference of Gaussian

Jin-Man Kwon*, Il-Ju Ko **, Dae-Sik Jang***

요 약

본 논문은 오디오 핑거프린트 데이터 생성 방법과 이를 이용한 오디오 데이터 비교 방법에 관한 것으로서, 오디오 데이터의 특징을 이용하여 음악을 식별하는 방법을 제시한다. 일반적으로 영상인식을 위해 많이 사용되는 가우시안의 차(Difference of Gaussian, DoG)를 오디오 데이터에 적용하여 음악이 급진적으로 변하는 부분을 추출하고, 해당 위치를 핑거프린트로 정의하는 방식이다. 이렇게 만들어진 핑거프린트는 음질의 변화에 민감하지 않으며, 음악 데이터의 일정 부분만으로도 원본과 동일 위치의 핑거프린트 추출이 가능하다. 이 시스템은 기존의 주파수 영역을 이용한 시스템 보다 오디오 핑거프린트의 데이터량과 계산량을 줄여줌으로써 검색을 할 때 보다 효율적인 성능을 나타낸다. 이를 응용하여 인터넷에 유통되는 복사된 음악의 저작권 보호, 또는 음악의 메타정보 등을 사용자에게 나타낼 수 있다.

Abstract

This paper, which is about the method of creating the audio fingerprint and comparing with the audio data, presents how to distinguish music using the characteristics of audio data. It is a process of applying the Difference of Gaussian (DoG: generally used for recognizing images) to the audio data, and to extract the music that changes radically, and to define the location of fingerprint. This fingerprint is made insensitive to the changes of sound, and is possible to extract the same location of original fingerprint with just a portion of music data. By reducing the data and calculation of fingerprint, this system indicates more efficiency than the pre-system which uses pre-frequency domain. Adopting this, it is possible to indicate the copyrighted music distributed in internet, or meta information of music to users.

▸ Keyword : 핑거프린트(fingerprint), 가우시안의 차(Difference of Gaussian, DoG), DTW

• 제1저자 : 권진만, 고일주 교신저자 : 장대식
• 투고일 : 2009. 11. 23, 심사일 : 2009. 11. 30, 게재확정일 : 2009. 12. 24.
* 숭실대학교 미디어학과 석사과정 ** 숭실대학교 미디어학과 교수 *** 군산대학교 컴퓨터정보공학과 교수

Ⅰ. 서 론

오디오 데이터를 검색하는데 있어서 검색 알고리즘은 매우 중요한 부분이다. 하지만 인식성능이 좋은 검색 알고리즘을 적용하더라도 검색이 되는 대상의 핑거프린트 데이터가 너무 많다면 검색 결과의 속도는 낮아지게 된다.

동일 음악을 검색하는데 있어서 해당 음악을 나타낼 수 있는 핑거프린트 정보와 이러한 핑거프린트 정보를 이용하여 유사한 핑거프린트를 검색하는 과정을 보면 핑거프린트의 개수는 검색 속도와 깊은 관련성을 가지고 있다.

본 논문의 목적은 해당 음악의 핑거프린트 정보를 최소한으로 추출하여, 동일 음악을 검색하는데 있어서 보다 적은 핑거프린트를 이용하여 효과적인 매칭 성공률을 보이는 것이다.

기존의 핑거프린트 알고리즘은 대부분 20ms 단위의 주파수 영역에서 데이터를 추출하여 핑거프린트를 정의하였다. 아무런 패턴 알고리즘 없이 약 4분의 음악데이터에서 핑거프린트를 추출하였을 경우 약 12000개의 핑거프린트가 추출된다. 이는 한 음악을 검색하는데 있어서 12000개의 핑거프린트를 매번 검사하게 된다. 만약 음악의 개수가 100곡으로 증가하게 되면, 핑거프린트 개수도 100배로 증가하게 되어 검색 속도는 점점 느려지게 된다. 이러한 문제점을 해결하기 위해 본 논문에서는 가우시안의 차(Difference of Gussian, DoG)를 이용한 알고리즘을 사용하였다.

DoG알고리즘은 이미지의 특정점을 추출하는 Scale Invariant Feature Transform(SIFT) 알고리즘에서 사용된다. SIFT 알고리즘은 이미지의 크기와 위치, 방향에 상관없이 같은 이미지를 추출할 수 있는 강점이 있다.[1]

그림 1의 (a)는 SIFT를 사용한 이미지 매칭결과이며, (b)는 SIFT에서 사용하는 DoG알고리즘을 응용하여 음악 데이터에 적용한 매칭결과이다. 즉, 이미지 데이터에서 특징점을 추출하는 방법과 같이 오디오 파형의 강·약이 되는 특징점을 추출하였다. 이러한 특징점들은 한 곡에서 약 1초에 1~3개의 핑거프린트를 만들 수 있으며, 4분의 음악데이터에서 대략 300개의 핑거프린트를 추출하고, 기존의 약 12000개의 핑거프린트 개수보다 40배 정도의 차이를 보였다.

기존의 알고리즘보다 핑거프린트 개수를 줄임으로써 검색 인덱스에서 처리할 데이터양을 줄이고, 인덱싱 속도를 높일 수 있다. 이는 동일한 메모리 영역에서 더욱 많은 오디오 데이터를 처리 할 수 있다. 그러므로 기존의 서비스에서 사용하는 동일한 메모리를 이용하여 더욱 많은 동일 음원 검색 및 인터넷에서 유통되는 음원의 저작권을 보호할 수 있다.

본 논문은 총 5장으로 구성되어 있으며, 각 장의 주요 내용은 다음과 같다. 2장에서는 본 논문에서 사용하는 알고리즘에 대한 설명과 기존의 오디오 핑거프린트 추출 방법에 대해 소개하며, 3장에서는 본 논문에서 제안하는 핑거프린트 추출 방법과 매칭 방법에 대해 기술한다. 4장에서는 본 논문에서 제안한 서로 다른 음질의 오디오 데이터를 비교하여 동일한 오디오 데이터를 추출한 실험 및 실험 환경에 대해 설명하고, 마지막으로 5장에서는 결론 및 향후 연구 과제에 대해 기술한다.

Ⅱ. 기존 연구

본 논문에서 사용하는 핑거프린트 알고리즘은 영상처리에서 주로 사용하는 알고리즘이다. 본 장에서는 오디오 핑거프린트에 대한 정의와 DoG알고리즘, DTW알고리즘에 대한 정의를 설명한다. 그리고 기존의 핑거프린트 알고리즘을 사용하는 멀티미디어 검색 서비스에 대해 소개한다.

2.1 오디오 핑거프린트 추출 알고리즘

오디오 핑거프린트는 일반적으로 오디오 데이터의 특징을 설명할 수 있는 데이터를 의미하는 것이다. 주로 주파수 변환 등의 방법에 의하여 오디오 데이터를 여러 가지 방법으로 분석하여 생성한다. 그리고 이를 이용하여 오디오 데이터의 무

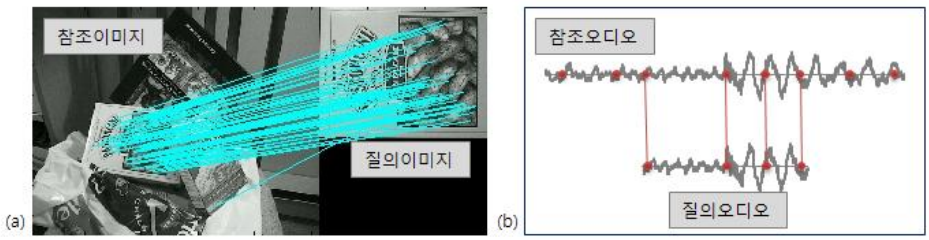

그림 1. (a) 이미지 매칭결과, (b) 오디오 매칭결과
Fig. 1. (a) Image matching results, (b) Audio matching results

단 도용 여부를 판별하거나 오디오 데이터를 검색하는 메타 정보 데이터로 사용한다.

본 논문에서는 핑거프린트를 추출하기 위해 DoG 알고리즘을 사용한다. DoG 알고리즘은 SIFT 알고리즘에서 키포인트를 탐색하는 알고리즘이다. 즉, 스케일 공간상에서 극대점, 혹은 극소점을 찾는 알고리즘이다. 이는 이미지의 스케일 변화나 이미지 변환에 있어서 같은 지점을 다시 탐색할 가능성이 높은 지점이다. 일반적으로 그라디언트(Gradient)나 헤시안(Hessain), 헤리스 코너(Harris corner) 함수와 같은 다른 이미지 함수와 비교하였을 때 보다 안정적인 이미지 특징을 만들어내는 것으로 알려져 있다.[2]

Lindeberg는 다양한 스케일공간을 생성하는 함수로 가우시안 함수를 사용하였으며, 함수 G는 가우시안 필터로 식 (2.1)과 같이 정의된다. 여기서 σ는 scale factor가 된다.[3]

$$G(x,y,\sigma)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2+y^2}{2\sigma^2}}. \quad \cdots\cdots (2.1)$$

식 (2.2)의 * 연산은 합성곱(Convolution)연산이며, 이미지 L은 원본 이미지 $I(x,y)$에 대하여 가우시안 필터를 적용하여 정의된다.

$$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y) \quad \cdots\cdots (2.2)$$

$$D(x,y,\sigma)=L(x,y,k\sigma)-L(x,y,\sigma). \quad \cdots\cdots (2.3)$$

식 (2.3)의 D는 가우시안 이미지의 차 연산 이미지로 일정한 배수(k)의 가우시안 필터가 적용된 이미지 사이의 차이로 정의된다.

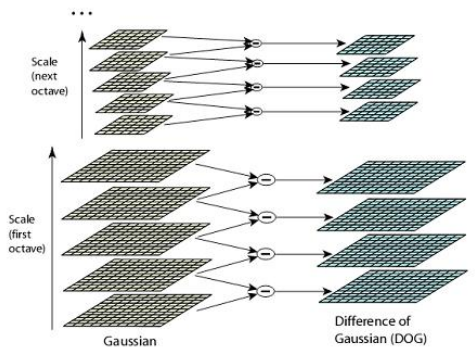

그림 2. 가우시안 차에 의한 영상
Fig. 2. Difference of Gaussian by image

그림 2는 가우시안 차 연산을 추출하는 방법으로 왼쪽의 이미지는 가우시안 필터가 적용된 이미지이며, 오른쪽의 이미지는 인접한 두 가우시안 이미지들의 차 연산을 통해 추출된 DoG 이미지이다.

그림 2에서 추출된 DoG 이미지를 이용하여 그림 3에서와 같이 극대값을 추출한다. 여기서 극대값은 극대점, 혹은 극소점이며, 이는 그림 상에서 'X' 표시가 되어있는 이미지를 현 DoG이미지로 가정하였을 때, 인접한 총 26개의 점들에 대하여 비교를 한다. 현 이미지의 'X'위치와 비교하여 'X'위치의 값이 가장 작거나 가장 큰 값을 가지는 점을 선택한다. 이를 반복하여 모든 DoG 이미지에서 극소점과 극대점을 추출하며, 이 지점을 SIFT에서는 이미지의 키포인트로 정의한다. 본 논문에서는 이러한 과정을 응용하여 오디오의 핑거프린트를 추출하게 된다. 이후 핑거프린트 간의 유사성을 추출하기 위해 본 논문에서는 DTW(Dynamic Time Warping) 알고리즘을 사용한다.

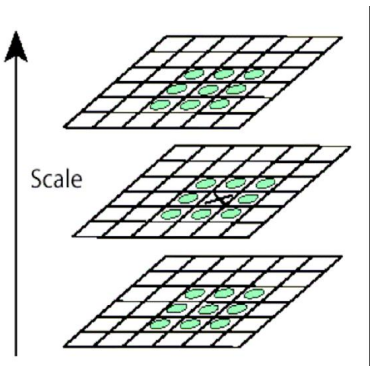

그림 3. DoG 이미지의 극대값 추출
Fig. 3. Extraction of the image peak

DTW 알고리즘은 음성과 같은 시계열 패턴에 사용되는 패턴 매칭 알고리즘의 하나로 두열의 각성분에 대한 거리척도 값을 비용으로 설정한다. 그리고 두열이 이루는 격자 상에서 각 열의 시작 성분에서 시작하여, 끝 성분에 이르기 까지 비용 테이블에 최소 비용을 순환적으로 선택하여 저장하는 점화식을 이용한다. 또한, 동적 계획법으로 매칭 함수를 찾아가면서 두열을 비교하는 알고리즘이다. 최종적으로 끝 성분에서 비용 테이블에 저장되는 비용 값이 두열에 대한 유사도가 된다.[4][5][6][7]

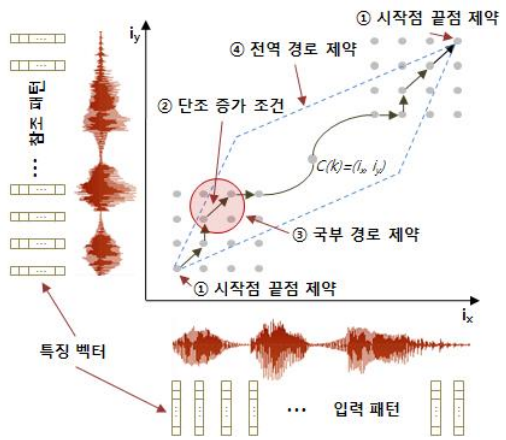

그림 4. DTW를 이용한 오디오 데이터의 유사도 비교
Fig. 4. Comparison to the similarity of the audio data using the DTW

그림 4에서와 같이 DTW는 5가지의 제약 조건을 가지고 있다. 첫 번째는 끝점 제약의 조건이다. 이는 입력 패턴과 참조 패턴의 시작점과 끝점을 일치시키고 비교가 이루어 져야 한다는 것이다. 두 번째는 단조 증가 제약 조건으로 최적 경로는 항상 단조로 증가해야 한다는 것이다. 세 번째는 국부 연속의 제약 조건이다. 이는 격자상의 한 노드에 도달하기 위한 경로에 제한을 둠으로 시간상에서 지나치게 수축되거나 팽창하는 것을 방지하기 위함이다. 네 번째는 전역 경로 제약 조건으로 탐색 시간을 줄이는 역할을 하며, 허용 가능한 경로 영역을 제한하는 조건이다. 마지막으로 기울기 가중치의 조건은 국부 경로의 비용을 계산할 때 모두 동일한 가중치를 주지 않고, 기울기에 따라서 서로 다른 가중치를 적용함으로써 시간에 비해 비합리적으로 불연속적 변화를 방지하는 것이다.

2.2 기존 시스템 연구

오디오 핑거프린트를 생성하는 기존의 방법으로는 여러 가지 방식이 제안되어 있으나, 검색하고자 하는 검색 대상 오디오 데이터의 양이 약 10000개 이상으로 많아질 경우 오디오 데이터의 핑거프린트를 생성하는 속도가 현저하게 느려지는 단점이 있다. 그로인해 많은 양의 오디오 데이터를 비교하는 경우에는 많은 문제점을 가지고 있다.

핑거프린팅은 인식에 유용한 성분을 오디오신호로부터 추출하는 과정이다. 핑거프린트는 일반적으로 정보의 압축, 차원 감소 과정과 관련된다. 핑거프린트에서는 이상적인 정답이 없기 때문에 오디오 인식을 위한 특징의 좋고 나쁨은 오디오 인식률로 판단된다.

핑거프린트의 주요 연구 분야는 인간의 청각특성을 반영하는 특징 표현, 다양한 잡음환경/화자/채널 변이에 강인한 특징추출, 시간적인 변화를 잘 표현하는 특징 등이 있다.[8]

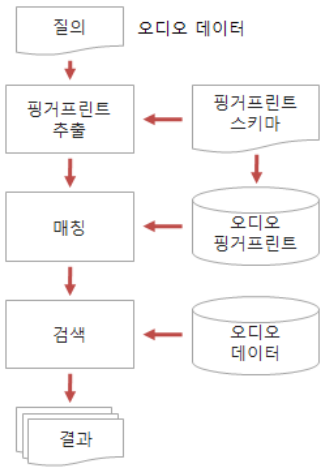

그림 5. 일반적인 오디오 검색 시스템
Fig. 5. Common audio search system

기존의 핑거프린트 추출방식은 오디오 데이터를 일정한 윈도우 영역으로 분류하여 프레임으로 정의한 뒤 FFT(Fast Fourier Transform) 알고리즘을 통하여 주파수 영역으로 변환한다. 이후 핑거프린트 스키마에 의해 특징별로 각각의 알고리즘을 거치게 되고, 최종 핑거프린트가 추출된다. 이렇게 추출된 핑거프린트 데이터들을 그림 5와 같이 DB의 오디오 핑거프린트 데이터와 질의 오디오 데이터의 핑거프린트와 서로 매칭하여 검색 결과를 출력한다.[9]

일반적인 핑거프린트 추출방식에서 사용되는 FFT과정처럼 주파수 영역으로 변환하는 알고리즘은 오디오 데이터를 특정 크기의 프레임으로 분할하여 모든 프레임을 복잡한 계산의 반복으로 처리하게 된다. 그러므로 시스템의 전체적인 효율성이 낮아지게 된다.

Wold는 "Muscle Fish"라는 시스템을 개발하여 소리를 크기(Loudness), 밝음(Brightness), 피치(Pitch), 음색(Timbre) 등의 지각 특징들을 통해 표현하였다. 그리고 Mahalanobis 거리 측정법과 Nearest Neighbor 규칙을 이용하여 핑거프린트 DB를 검색하였다. 그러나 이 시스템은 악기 소리, 동물 소리 그리고 환경 소리 등의 음향 효과 음에 국한되어 음악 데이터를 사용하는 시스템에서는 한계가 있었다.[10]

Wang은 이진 특징을 사용하는 오디오 유전자 추출 방식을 사용한다. 이 알고리즘을 이용하여 데이터베이스에 있는 각 오디오 데이터를 각 프레임마다 스펙트럼의 에너지의 32

개 주파수 대역별로 0 또는 1의 값으로 표현하고, 이 값을 검색 테이블 값에 (오디오 신호 ID, 해당 프레임 i)로 추가 등록한다.[11]

검색의 경우 임의의 수 초 구간의 입력 오디오에 대해 동일한 방식으로 32비트 패턴을 추출하여 이 테이블을 검색하는 방식이다. 이 알고리즘은 검색 테이블의 각 엔트리에 등록된 (오디오 신호 ID, 프레임 index)의 개수가 가변적이고 충분한 검색 속도를 보장할 수 없다는 단점이 있다. 또한 바이너리 특징벡터 추출방식이 고정적이어서 입력 신호에 발생한 손상, 즉 오디오 데이터의 음질에 상대적으로 취약하다.

Google에서는 음악의 특성을 나타내는 파형의 패턴을 분석하여 오디오 핑거프린트를 추출했다. 이를 Waveprint라고 정의하고 있으며, 해당 음원을 나타내는 핑거프린트들의 집합이다.[12]

Google의 알고리즘은 인덱싱, 변환, 매칭 총 3단계로 구분한다. 첫 단계인 인덱싱 단계에서는 음원들의 파형패턴을 잘게 쪼개어 부분 데이터를 생성하고, Waveprint를 추출한다. 두 번째 단계인 변환 단계에서는 질의 오디오 데이터에 대해 Waveprint 형태로 변환하며, 마지막 매칭 단계에서 입력된 Waveprint를 DB의 Waveprint와 비교한다.

Google의 핵심 알고리즘은 Wavelet을 기반으로 하고 있기 때문에 Waveprint를 확대, 축소하여도 그 고유한 패턴은 변하지 않는다.[13] 즉, 음악의 속도가 빨라지거나 느려져도 동일한 패턴을 추출 할 수 있는 장점이 있다.

국내 동영상 서비스인 엔써미는 오디오 데이터를 MFCC나 PLPC 또는 LPC 중 하나 이상의 조합을 사용하여 특징벡터를 추출한다. 이러한 특징 분포 데이터를 이용하여 제 1 프레임을 구성하고, 각각에 대한 특징 분포 데이터의 빈도를 계산함으로써 제 2프레임을 구성한다. 여기서 특징 분포 데이터는 특징 데이터의 분포 특성을 나타내는 데이터로 코드북(codebook)을 참조하여 생성한다. 코드북은 다수의 오디오 데이터 특징 벡터들을 미리 추출하여 k-means와 같은 클러스터링 알고리즘을 통해 이들의 벡터 공간에 분포시키고, 벡터 공간 상에서 특징 벡터들을 그룹화 한다. 이후 각각의 그룹에 포함되어 있는 특징 벡터들의 평균값을 계산한다. 계산된 평균값과 각 그룹에 대한 인덱스값을 저장하고 있는 데이터를 이용하여 핑거프린트를 구성한다.[14]

엔써미에서 사용하는 알고리즘은 기존의 오디오 핑거프린트 생성 방식에 비하여 속도가 현저하게 개선되었지만, MFCC나 LPC같은 알고리즘을 사용하기 때문에 오디오 데이터의 음질에 취약한 단점을 가지고 있다. 하지만 Google의 알고리즘은 2GB의 메모리에서 약 31,000곡(3분 기준)의 오디오 데이터를 처리할 수 있고, 엔써미의 알고리즘은 동일 메모리에서 약 300,000곡의 오디오 데이터를 처리 할 수 있다. 이는 핑거프린트의 개수를 줄임으로써 검색의 인덱스 정보를 줄이는 장점이 있다.

대부분의 기존 연구들은 오디오 핑거프린트를 추출하고 검색하기 위하여 특징 벡터 추출 및 패턴 인식 기법에 관한 연구들이 주를 이루고 있다. 하지만 대부분 알고리즘은 주파수 영역에서 오디오 핑거프린트를 추출한다. 또한, 오디오 데이터의 전 영역으로 부터 주파수 변환 알고리즘을 사용하기 때문에 많은 계산을 요구한다.

III. DoG와 DTW를 이용한 오디오 데이터 검색

기존의 핑거프린트 방식들은 대부분 주파수 영역에서 추출되는 방식을 취하고 있다. 주파수 영역으로 변환된 데이터는 음악 데이터의 주파수 성향을 분석할 수는 있지만, 동시에 시간에 대한 정보를 잃어버리게 된다. 시간의 정보가 없기 때문에 모든 음악데이터를 주파수 영역으로 변환해야 한다. 그로 인해 불필요한 정보인 무음부분, 또는 소리가 작고 약한 부분에서도 주파수를 추출함으로써 핑거프린트의 데이터양이 많아지게 된다. 이러한 단점을 보완하기 위해 본 논문에서는 웨이브 파형의 특징을 추출하여 음질에 무관하고 항상 동일한 위치에서 추출되는 핑거프린트를 정의하였다.

본 논문의 핑거프린트 시스템은 두 단계로 정의 할 수 있다. 첫 단계는 음악 데이터로부터 최소한의 핑거프린트를 추출한다. 이후 두 번째 단계에서 질의 데이터의 핑거프린트와 DB에 저장되어 있는 참조 핑거프린트와의 순차검색을 통해 가장 매칭율이 좋은 매칭 지점 추출한다.

3.1 핑거프린트 추출

본 논문의 핑거프린트 추출과정은 그림 6에서와 같이 전처리과정을 포함하여 특징점 추출과 지역정보 추출, 핑거프린트 정의로 분류 할 수 있다. 핑거프린트 추출과정의 첫 단계인 전처리 과정에서는 입력되는 오디오 데이터의 스테레오 정보를 모노 데이터로 변환하고, 잡음을 제거하기 위해 가우시안 필터를 거치게 된다. 이후 특징점 추출단계에서는 DoG 알고리즘을 사용하여 급진적으로 변하는 오디오 데이터의 극대값을 추출한다. 이후 지역정보 추출단계에서 프레임 단위의 극대값을 이용하여 특징점의 위치를 추출하게 된다. 마지막 핑거프린트 정의단계에서는 추출된 특징점 위치에서 FFT 알고

리즘을 적용하여 특징점 주변의 지역적인 정보를 추출한다. 그리고 특징점 위치 정보와 5차수의 FFT계수를 이용하여 총 6차수의 핑거프린트를 정의한다.

그림 6. 핑거프린트 추출과정
Fig. 6. Fingerprint extraction process

핑거프린트를 추출하기 위해 입력되는 오디오 데이터는 DB에 저장되어 있는 참조데이터와 음질이 다를 수 있다. 또는 오디오 데이터의 전체 부분이 아니 일부분만 입력될 수도 있다. 때문에 핑거프린트는 음질 혹은 노이즈에 강인해야 하며, 오디오의 시작 위치가 다르더라도 항상 같은 핑거프린트가 추출되어야한다.

본 논문은 *DoG*의 극대값을 이용하여 시작 위치와 무관한 핑거프린트 위치 점을 추출하였다. 또한 기존의 주파수 분석을 통한 오디오 데이터 분석은 해당 오디오 데이터 음질의 변화에 민감하기 때문에 주파수 분석을 사용하지 않고 오디오 파형의 패턴을 이용하여 음질에 강인한 핑거프린트 위치를 추출하였다. 이후 추출된 핑거프린트 위치에서 일정 영역의 주파수 분석을 통해 핑거프린트 위치의 지역적인 정보를 추가하였다. 이러한 과정은 오디오 데이터의 음질, 또는 오디오 데이터의 일부분이 입력되더라도 강인한 특징점을 추출 할 수 있다.

3.1.1 가우시안의 차를 이용한 극대값추출

핑거프린트 추출에 있어서 첫 단계는 *DoG*알고리즘을 통해 파형의 패턴 변화를 나타내 주는 극대값을 추출하는 단계이며, 그림 7은 전처리를 거친 오디오 데이터에서 *DoG*데이터를 추출하는 과정이다.

본 논문에서 사용되어진 DoG 알고리즘은 이미지의 특징점을 추출하는 알고리즘을 오디오의 특징을 추출하는 알고리즘으로 변형하였다. 오디오의 특징점을 추출할 수 있도록 변경한 DoG 알고리즘은 부드럽게 변화하는 오디오 데이터의 웨이브 파형에서 특징점을 추출하지 않고, 급진적으로 변화하는 웨이브 파형에서만 특징점을 추출한다.

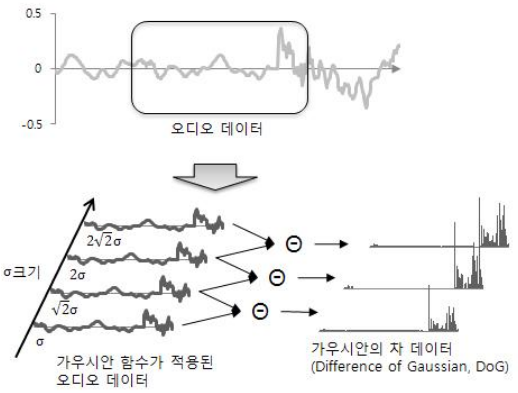

그림 7. DoG를 이용한 극값 추출과정
Fig. 7. Peaks extraction process using the DoG

일반적으로 DoG 알고리즘에서 사용하는 가우시안 커널 함수는 이미지를 처리하기 위해 2차원의 가우시안 커널 함수를 사용한다. 하지만 오디오 데이터에서는 이미지 데이터와는 달리 1차원 형식의 데이터를 사용하기 때문에 식(3.1)과 같은 1차원 가우시안 커널 함수를 사용한다.

$$G(x,\sigma)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2}{2\sigma^2}}. \quad \cdots\cdots (3.1)$$

가우시안 함수가 적용된 오디오 데이터는 식 (3.2)와 같이 가우시안 함수의 σ값을 이용하여 $L(x,\sigma)$를 추출해야 한다. 이는 일정한 크기의 가우시안 함수 $G(x,\sigma)$에 오디오 데이터 $A(x)$를 적용하여 합성곱(Convolution)을 통해 가우시안 함수가 적용된 오디오 데이터를 추출하게 된다.

$$L(x,\sigma)= G(x,\sigma)*A(x) \quad \cdots\cdots (3.2)$$

각각의 가우시안 함수가 적용된 오디오 데이터에서 DoG 값을 구하기 위해서는 k값을 적용한 $L(x,k\sigma)$값과 $L(x,\sigma)$값의 차를 구하여 식 (3.3)과 같이 DoG 데이터를 추출하게 된다. 이 값을 $D(x,\sigma)$으로 정의한다. 본 논문에서 k값은 $\sqrt{2}$로 적용하여, 순차적으로 σ, $\sqrt{2}\sigma$, 2σ, $2\sqrt{2}\sigma\cdots$ 의 $k\sigma$을 적용하였다.

$$\begin{aligned} D(x,\sigma) &= (G(x,k\sigma)-G(x,\sigma))*A(x) \quad \cdots\cdots (3.3) \\ &= L(x,k\sigma)-L(x,\sigma). \end{aligned}$$

추출된 *DoG* 데이터인 $D(x,\sigma)$를 살펴보면 웨이브 파형에 있어서 파형이 급격히 변하거나 패턴의 형식이 변화하는 부분

에서 높은 *DoG*값이 추출된다. 이는 이미지에서의 *DoG*값은 Edge정보를 가지게 되지만, 오디오 데이터에서는 오디오의 파형이 급격하게 변화하는 부분을 나타낸다고 할 수 있다. 또한 이 값은 오디오의 음질이 변하여도 동일한 위치에서 동일하게 높은 DoG 값이 추출된다.

하나의 DoG 데이터는 두 개의 가우시안 함수가 적용된 오디오 데이터를 이용하여 추출한다. 즉, $D(x,\sigma)$를 추출하기 위해서는 $L(x,\sqrt{2}\sigma)$와 $L(x,\sigma)$의 차를 계산하며, $D(x,\sigma)$를 추출하기 위해서는 $L(x,2\sigma)$와 $L(x,\sqrt{2}\sigma)$의 차를 계산한다. 이는 동일한 오디오 데이터를 서로 다른 가우시안 함수를 두 번 적용하여 추출하여야 한다. 이러한 과정을 줄이기 위해 식 (3.4)와 같이 가우시안 함수 $G(x,\sigma)$를 미리 추출하여 $G(x,k\sigma)$와 $G(x,\sigma)$의 차를 먼저 구한 뒤 오디오 데이터 $A(x)$를 추출 할 수 있다.

$$DoG \approx G(x,k\sigma) - G(x,\sigma) \quad \cdots\cdots\cdots\cdots \quad (3.4)$$
$$= \frac{1}{\sqrt{2\pi}}\left[\frac{1}{k\sigma}e^{-\frac{x^2}{2k\sigma^2}} - \frac{1}{\sigma}e^{-\frac{x^2}{2\sigma^2}}\right].$$

추출된 DoG 데이터를 이용하여 극대값을 가지는 DoG 데이터의 위치를 추출하게 된다.

3.1.2 핑거프린트의 위치 추출

오디오 데이터의 DoG데이터는 이미지의 DoG데이터와는 달리 하나의 음을 표현하기 위해 많은 데이터를 사용한다. 이는 이미지의 1픽셀정보는 하나의 점을 나타내며, 미세하지만 사람이 인식할 수 있는 정보이다. 하지만 오디오 데이터의 1포인트 정보는 사람이 인식할 수 없는 매우 작은 데이터이며, 사람이 오디오를 인식하기 위해서는 최소 20ms의 오디오 데이터가 필요하다. 이를 계산하면, 초당 44.1KHz로 샘플링 되어 있는 오디오 데이터에서 약 2048 포인트로 표현할 수 있다. 이러한 소리의 특징을 이용하여 인간이 인식할 수 있는 소리가 음악의 특징을 정의 할 수 있는 소리와 같다는 가정으로 본 논문에서는 오디오 데이터의 전후 10ms 단위의 데이터로 분리하여 합계를 통한 하나의 DoG 프레임을 구성하였다.[15]

그림 8은 핑거프린트 첫 단계에서 추출된 3개의 DoG 데이터를 이용하여 극대값을 추출하는 과정이다. 기존의 DoG 알고리즘과 같이 중간 영역의 프레임(e)을 선택하여 (a), (b), (c), (d), (f), (g), (h), (i)의 프레임과 비교하게 된다. 이때, 중심 프레임(e)의 값이 가장 높을 경우, 이를 핑거프린트 지점으로 정의한다.

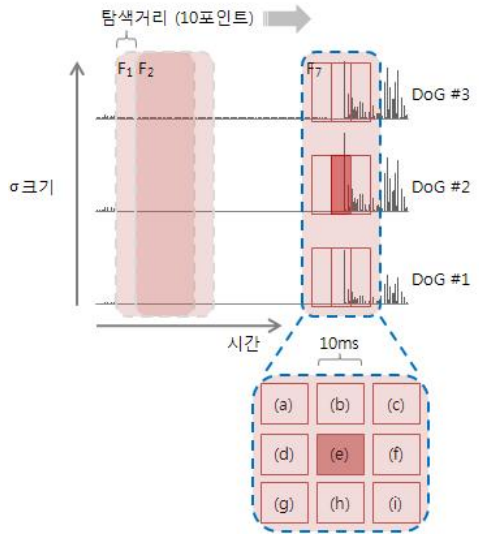

그림 8. 핑거프린트 위치 추출과정
Fig. 8. The location of the fingerprint extraction process

중심 프레임은 DoG 데이터의 시간상에서 순차적으로 선택되어 추출한다. 즉, 그림 8에서 각각의 프레임셋 F_1, F_2 … 와 같이 10포인트씩 프레임셋 F_n을 이동하여 중심 프레임(e)를 추출하게 된다. 이때, 오디오 데이터의 파형이 자주 급격히 변화하는 댄스나 락과 같은 비트가 빠른 음악이라면 극대값이 높은 중심프레임들이 자주 추출하게 된다. 이를 해결하기 그림 9과 같이 1초에 해당하는 윈도우를 적용하여 해당 윈도우에서 극값이 가장 높은 하나의 핑거프린트 지점만을 추출한다. 이러한 제한적인 윈도우 처리로 극대값의 출현범위를 초당 1~2개로 줄일 수 있다. 또한, 제한적인 변수를 사용하지 않고도 개수를 축소할 수 있다.

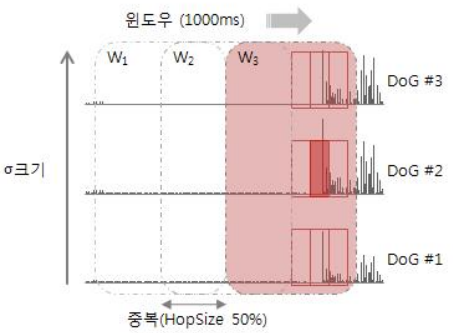

그림 9. 윈도우를 통한 프레임 검색
Fig. 9. Search through the window frame

하나의 윈도우는 핑거프린트 위치를 추출 할 수 있는 영역에 대한 공간을 제한한다. 또한 50%의 중첩을 적용하여 윈도우를 이동하고, 해당 윈도우 내에서 중심 프레임을 생성한다. 중심 프레임과 주변의 프레임을 비교하여 극대값이 가장 높고, 해당 윈도우 내에서 가장 높은 극대값을 가지고 있다면, 해당 중심 프레임을 소속되어있는 윈도우의 핑거프린트 위치로 정의한다. 이때 중복되는 핑거프린트 지점이 추출된다면 같은 핑거프린트 지점으로 정의한다. 이러한 과정을 통해 결과적으로 1초의 데이터에서 약 1~3개 정도의 핑거프린트를 추출 할 수 있다.

이후 추출된 핑거프린트 위치 정보를 이용하여 지역적인 정보와 함께 하나의 핑거프린트를 정의하게 된다. SIFT 알고리즘에서는 특징점 주변의 픽셀 방향을 이용하여 지역적인 정보를 추출하였다. 하지만 오디오 데이터의 파형 데이터는 방향성분을 가지고 있지 않기 때문에 파형의 주파수 정보를 추출하는 FFT를 이용하여 지역적인 정보를 추출한다.

3.1.3 FFT를 이용한 지역정보 추출

핑거프린트 추출의 마지막 단계로 추출된 핑거프린트 위치에 대한 지역적인 정보를 추출한다. 이 지역적인 정보는 핑거프린트의 위치를 식별하기 위한 과정이다.

DoG 데이터에서 추출된 핑거프린트 위치에서 2048포인트 단위의 고속푸리에변환(Fast Fourier Transform, FFT)을 적용하여 주파수 성분을 추출하였다. FFT는 함수의 근사값을 계산하는 알고리즘이다. 이 알고리즘은 푸리에 변환에 근거하여 근사공식을 이용한 이산푸리에변환(Discrete Fourier Transform)을 계산 할 때 연산횟수를 줄일 수 있다.

다양한 음질의 동일한 오디오 데이터에서 FFT함수의 근사값은 서로 유사한 특성을 가지고 있다. 따라서 음질의 차이가 있더라도 동일한 웨이브 파형의 패턴일 경우, FFT함수의 근사값은 저차원 일수록 서로 비슷한 값을 가지고 있다.

그림 10은 동일한 음원에 대하여 128k와 192k, 256k, 320k의 음질로 변환하여 일정한 구간을 FFT처리하여 출력한 결과이다. 그래프에서와 같이 저주파수에서 중주파수 영역에 대해서는 서로 유사한 값을 보여준다. 그러나 고주파수 영역으로 이동 할수록 데이터가 서로 상이한 결과를 보인다. 이는 음질의 차이는 주로 고주파수 영역에서 많이 발생되기 때문이다. 또한, 일반적인 오디오 데이터의 중요 데이터는 저주파수 영역에 포함되어 있다. 1 ~201의 FFT Index는 약 0~2100Hz정도를 나타내며, 오디오의 중주파수 영역까지를 포함하고 있다.[16] 따라서 본 논문에서는 음질의 영향을 많이 받지 않는 저주파수 영역에서 중주파수 영역까지의 데이터를 사용하여 지역적인 정보를 추출하였다.

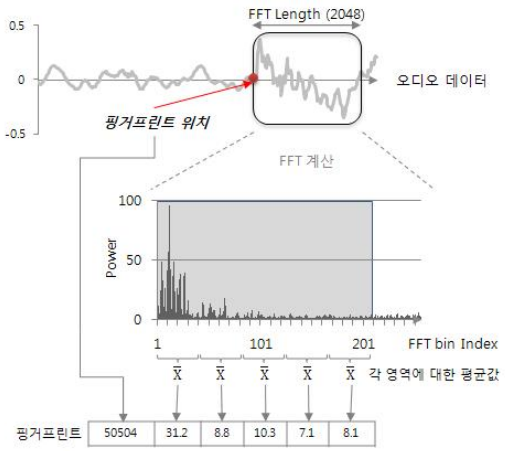

그림 11. FFT를 이용한 지역정보 추출과정
Fig. 11. Local information extraction process using the FFT

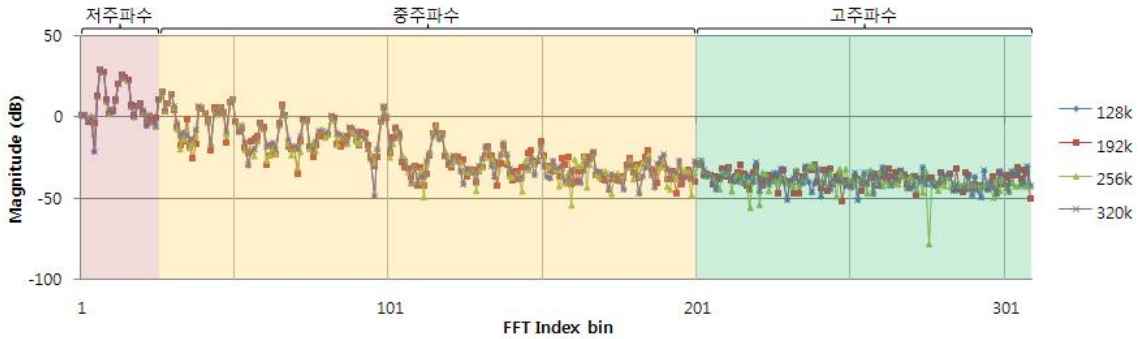

그림 10. FFT 비교
Fig. 10. FFT comparison

그림 11은 하나의 핑거프린트 지점으로부터 핑거프린트 지역 정보를 추출하는 전체적인 과정을 보여주고 있다. 그림 8에서 추출된 특징 점으로 부터 2048 크기로 FFT를 추출하고, 1~201까지의 FFT Index bin 데이터를 사용한다.

핑거프린트의 지역정보는 FFT를 통해 추출된 200개의 FFT Index bin을 5등분한다. 그리고 각각의 영역에 대한 평균값($\bar{x}$)을 추출하여 주파수 영역별 특징 정보를 정의한다. 그리고 DoG 평균값을 통해 추출된 특징점의 위치와 각 영역의 평균값을 통해 추출된 5개의 FFT 특징을 이용하여 하나의 핑거프린트를 구성하게 된다.

표 1. 핑거프린트 위치와 지역정보에 따른 매칭율
Table 1. Matching rate based on Fingerprint location and local information

	특징점 위치	지역정보 (FFT)	특징점위치+ 지역정보(FFT)
매칭율	97.44%	93.37%	91.02%

표 1은 핑거프린트의 위치값만을 사용한 매칭율과 지역 정보인 FFT만을 사용한 매칭율, 그리고 핑거프린트의 위치와 지역정보 FFT를 같이 사용하여 매칭한 결과를 보여주고 있다. 간단히 핑거프린트의 위치만을 사용한 매칭 방법이 더 좋은 매칭율을 보인다. 하지만 많은 오디오 데이터의 매칭에서는 중복 에러를 발생할 수 있는 확률이 높아진다. 이는 특징점 위치 정보만으로는 비슷한 정보를 가지고 있는 오디오 데이터가 많기 때문이다. 이를 보완하기 위해 매칭율은 낮지만 많은 오디오 데이터에서 중복 에러를 피할 수 있기 때문에 본 논문에서는 핑거프린트의 위치 값과 FFT를 동시에 사용한다.

3.2 핑거프린트 매칭

핑거프린트 매칭과정은 각각의 음원에서 추출된 핑거프린트 정보에 대한 매칭율을 평가하기 위한 과정이다. 매칭율을 평가하기 위해 DTW 알고리즘을 사용하며, 질의 데이터와 참조 데이터간의 유사도를 계산하여 가장 높은 유사도를 추출한다. 여기서 유사도가 가장 높은 부분은 질의 데이터를 기준으로 참조 데이터와의 매칭율이 가장 높은 부분으로 유사도에 따라 매칭위치가 결정된다.

매칭과정에서 사용되는 질의 데이터는 검색하고자 하는 음원의 일부 또는 전체의 핑거프린트 정보이다. 이는 음원의 특정부분 즉, 10초 구간이나 30초 구간으로 임의의 위치에서 추출될 수 있다. 그리고 참조 데이터는 검색대상이 되는 핑거프린트 정보로 해당 음원에 대한 전체적인 내용을 가지고 있다.

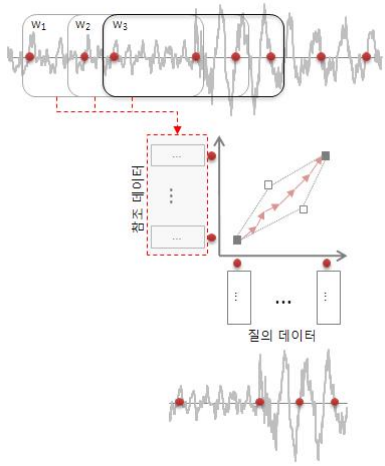

그림 12. 핑거프린트 매칭 과정
Fig. 12. The fingerprint matching process

본 논문에서 사용되는 DTW 알고리즘의 참조 데이터는 DTW알고리즘의 끝점제약조건으로 인해 전체의 핑거프린트 정보를 한 번에 사용하지 않고, 그림 12와 같이 질의 데이터의 핑거프린트 개수만큼 분할한다. 즉, 참조 데이터를 질의 데이터의 개수만큼 윈도우를 적용하여 $W_i = \{W_1, W_2, W_3 \sim W_I\}$로 나누고, 순차적으로 질의 데이터의 위도우인 W_q와 비교하는 것이다. 여기서 참조 데이터의 전체 윈도우 개수인 I값은 $n(W_I) - n(W_q) + 1$로 정의할 수 있다. 예로 참조 데이터의 핑거프린트가 300개이고, 질의 데이터의 핑거프린트가 20개라면, 참조 데이터를 (300-20+1=281)개의 윈도우로 나눈다. 그리고 하나의 윈도우에 속하는 핑거프린트 개수를 20개로 정의한다. 이후 순차적으로 참조 데이터와 질의 데이터에서 각각 20개의 핑거프린트를 이용하여 유사도를 측정한다.

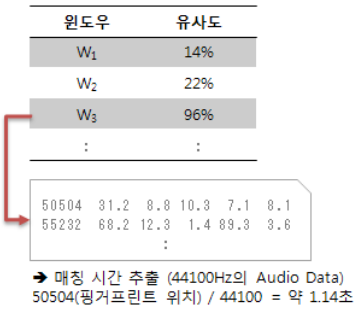

그림 13. 매칭 지점 추출
Fig. 13. Matching point extraction

그림 13은 참조 데이터의 윈도우에 따른 질의 데이터에 대한 유사도를 나타낸다. 즉, DTW과정을 통해 추출된 유사도 정보를 나열하여 유사도가 가장 높은 W_3 윈도우 지점을 질의 데이터와 매칭되는 매칭 위치로 정의하고, 해당 윈도우의 핑거프린트 위치를 통해 매칭지점의 시간을 추출하게 된다.

IV. 실험 및 결과

본 논문의 실험에서는 제안한 알고리즘의 파라미터 값에 대한 실험과 이를 이용하여 기존의 핑거프린트 알고리즘을 통한 데이터양을 비교하는 실험을 수행한다.

4.1 실험환경 및 실험데이터

실험을 위한 컴퓨터 환경은 펜티엄4, 3GHz와 2GByte의 메모리에서 실험하며, 알고리즘은 Microsoft Visual Studio 2008을 사용하여 C++언어로 구현한다.

실험의 샘플 데이터로는 Ballad 50곡, Dance 50곡, jazz 50곡, Rock 50곡 총 200곡을 선정했다. 여기서 장르를 구분하는 이유는 음악 데이터의 특성에 따른 핑거프린트 데이터양을 비교하기 위함이다. 그리고 음악 데이터를 128k, 192k, 256k, 320k로 구분하여 음악의 음질에 따른 핑거프린트를 비교한다.

질의 데이터인 부분 오디오 데이터는 각 음악을 30초, 60초, 120초 위치에서 10초, 30초, 60초 구간으로 추출하여 사용한다. 이는 질의 데이터에 대하여 길이에 비례한 참조 데이터와의 매칭율을 검증하기 위함이다. 실험에 사용된 모든 오디오 파일은 스테레오, 16bit, 44100Hz의 MP3파일로 구성되어 있다.

4.2 파라미터 설정

본 논문에서 사용하는 파라미터로는 DoG데이터의 개수를 지정하기 위한 가우시안 함수의 개수와 극대값을 추출하기 위한 DoG데이터의 윈도우 범위로 두 가지가 있다.

DoG 데이터의 개수는 가우시안 함수의 개수로 결정된다. SIFT에서 사용하는 가우시안 함수의 개수가 증가하면, 극값으로 추출되는 데이터의 양 또한 증가한다. 이는 DoG 이미지의 개수가 증가함으로써 DoG 이미지에 대한 후보 키포인트 데이터 또한 증가하기 때문이다. 하지만 본 논문에서는 일정한 영역의 윈도우를 설정하여 DoG 프레임에 대한 최대값을 추출하기 때문에 후보 키포인트는 하나만 나오게 된다.

가우시안 함수를 추출하기 위한 $k\sigma$값은 가우시안 개수에 따라 순차적으로 σ, $\sqrt{2}\sigma$, 2σ, $2\sqrt{2}\sigma\cdots$ 으로 적용된다. $k\sigma$ 값이 증가할수록 가우시안 함수가 적용된 오디오 데이터의 파형에 대한 변화가 작아짐으로 DoG에서 추출되는 값 또한 작아지게 된다. 따라서 각각 윈도우에 대한 DoG 프레임의 최대값은 대부분 σ, $\sqrt{2}\sigma$, 2σ, $2\sqrt{2}\sigma$의 $k\sigma$값으로 생성된 가우시안 함수로 생성된 DoG 프레임에서 추출된다.

그림 14는 본 논문의 알고리즘에서 다양한 가우시안 개수를 적용하여 각각의 처리속도를 비교한 그래프이다. 처리속도에 대한 비교 기준으로 최소한의 가우시안 개수인 4를 지정하였다. 이는 3개의 DoG 데이터를 추출하기 위함이다. 또한 하나의 극대값을 추출하기 위해서는 최소 3개의 DoG데이터가 필요하기 때문이다.

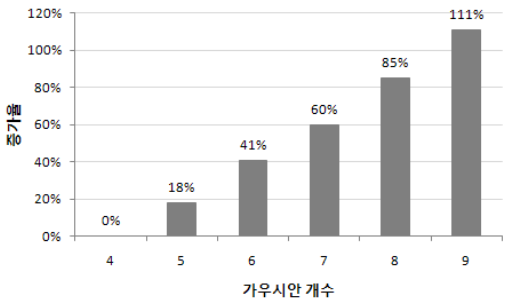

그림 14. 가우시안 개수에 따른 처리속도 증가율
Fig. 14. Growth rates based on the number of Gaussian process

그림 14의 그래프를 보면 가우시안 개수가 증가하면서 그에 따른 처리속도 또한 증가하게 된다. 가우시안 개수가 증가하게 되면 가우시안 함수가 적용된 오디오 데이터를 가우시안 개수만큼 생성하기 때문이다. 하지만 가우시안 개수가 증가하더라도 윈도우의 최대값은 하나의 DoG 프레임만을 추출하기 때문에 핑거프린트의 매칭율은 증가하지 않는다. 또한 추출되는 핑거프린트 정보 또한 동일하다. 그렇기 때문에 본 논문에서는 최소 가우시안 함수 개수인 4개의 가우시안 함수를 사용한다.

그림 15는 DoG 프레임의 최대값을 추출하기 위한 윈도우의 크기를 실험한 그래프이다. 실험에서 사용한 윈도우의 크기는 500ms와 1000ms, 1500ms, 2000ms 영역으로 정의한다. 각 윈도우 크기별로 실험한 결과 윈도우의 크기가 증가할수록 핑거프린트의 개수는 감소하게 된다. 또한, 핑거프린트의 개수가 감소할수록 매칭율이 줄어든다. 이는 비교 대상이 되는 핑거프린트의 개수가 작기 때문이다. 그러나 윈도우의 크기가 작으면 핑거프린트의 개수는 증가하게 되어 핑거프린트의 용량이 증가하게 된다. 본 논문에서는 핑거프린트의

용량과 매칭율을 고려하여 1000ms의 윈도우 크기로 지정한다. 이는 약 4분의 오디오 데이터에서 1000ms의 윈도우 크기에 따른 매칭율은 평균 90%의 매칭율을 보인다. 또한 약 300개 정도의 핑거프린트 개수를 추출한다.

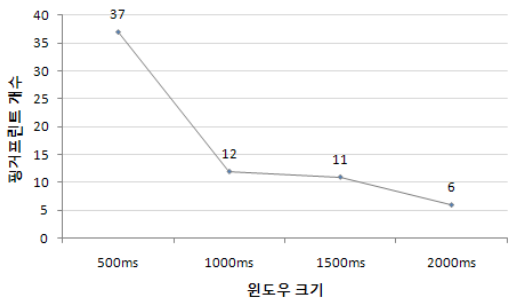

그림 15. 윈도우 크기에 따른 DoG 데이터의 개수
Fig. 15. Number of DoG Data based on the window length

3.3. 시스템 성능

본 논문의 실험은 4개의 가우시안 함수와 1000ms의 윈도우를 사용하여 매칭율을 계산한다. 매칭율은 질의 데이터의 핑거프린트 개수를 기준으로 참조 데이터와 매칭되는 개수를 비교한다.

$$M(W_i) = \frac{|n(W_q) \cap n(W_i)|}{n(W_q)}, (i = 1,2,\cdots,I) \quad \cdots\cdots\cdots\cdots (4.1)$$

식 (4.1)은 본 논문에서 매칭율을 추출하기 위한 식이다. 본문 3.2에서 정의한 바와 같이 $n(W_q)$은 질의 데이터의 핑거프린트의 개수를 나타낸다. 그리고 I는 참조 데이터의 핑거프린트 개수를 질의 데이터의 핑거프린트 개수로 나누어 총 비교 대상이 되는 참조 핑거프린트의 윈도우 개수를 나타낸다. 또한, $n(W_i)$은 W번째의 참조 데이터 핑거프린트 개수를 나타낸다. 즉, 질의 데이터의 핑거프린트 개수인 $n(W_q)$를 기준으로 매칭율 $M(W_i)$를 정의하게 된다.

최종 매칭 지점은 참조 데이터에서 각각의 윈도우 유사도를 추출하여 유사도가 가장 높은 윈도우를 선택하고, 매칭 지점을 추출한다.

표 2. 음악 종류별 평균 핑거프린트 개수
Table 2. The average number of different types of music fingerprint

	Ballad	Dance	Jazz	Rock
평균 핑거프린트 개수	298	278	303	307

표 2는 약 4분의 오디오 데이터에서 각각 장르에 따라 추출되는 핑거프린트의 개수를 비교한 표이다. 본 논문의 알고리즘은 음악의 파형 특성을 이용한 알고리즘이다. 따라서 파형의 특성에 따른 핑거프린트 개수를 추출하기 위해 조용한 음악에 대하여 추출되는 핑거프린트의 개수를 비교한다. 비교 결과 모든 장르에 대하여 평균 300개 정도의 핑거프린트를 추출하였다. 또한, 평균 90%의 매칭율로 오디오 데이터의 장르에 따른 차이가 없다.

표 3. 질의 오디오 데이터의 시간 양에 따른 매칭율
Table 3. A matching rate based on the amount of time of a query audio data

음질	10's	30's	60's
원본(128k)	94%	100%	100%
192k	86%	92%	100%
256k	90%	94%	100%
320k	90%	93%	100%

표 3는 128k의 오디오 데이터를 참조 데이터로 하여 각 음질별로 실험을 한 데이터이다. 또한 질의 데이터를 10초, 30초, 60초의 부분 오디오 데이터로 정의하여 질의 데이터의 시간에 따른 매칭율을 나타낸다. 이때의 참조 데이터와 질의 데이터의 매칭 범위는 ±2초로 정의한다. 즉, 참조 데이터의 매칭 위치가 120초 지점일때, 118초부터 122초 까지는 같은 매칭 지점으로 판단한다. 이는 음원에 있어서 ±2초의 차이는 일반적인 사람에 있어서 민감하게 반응하지 않기 때문이다. 실험 결과를 보면 입력 데이터의 시간 양이 많아질수록 매칭율이 높아진다. 이는 비교대상이 되는 핑거프린트의 개수가 많기 때문이다.

표 4. 다른 시스템과의 데이터 사용량에 대한 비교
Table 4. Comparison with other systems for data-intensive

	1곡당 메모리 사용량	2GB 메모리
MFCC (13차)	약 234KByte	약 8,000곡
구글 (Wavelet 기반)	약 64KByte	약 31,000 곡
엔써미 (MFCC 기반)	약 6KByte	약 300,000 곡
제안방법 (DoG 기반)	약 3KByte	약 600,000 곡

본 논문에서 제안하는 알고리즘은 기존의 핑거프린트 시스템보다 낮은 메모리 사용량으로 평균 90%의 매칭율을 보인다. 표 4는 구글 및 동영상 검색 서비스인 엔써미에 대한 비교 테이블이다. 구글과 엔써미 모두 주파수 영역에서의 데이터를 기반으로 하고 있으며, 각각 2GB 메모리에서 31,000곡과 300,000곡의 데이터를 저장 할 수 있다. 이에 반에 본 논문에서 제안하는 알고리즘은 하나의 핑거프린트 용량을 10Byte로 정의하였을 때, 총 2GB의 메모리에서 약 600,000곡의 오디오 데이터를 저장 할 수 있다.

본 실험을 통해 제안하는 알고리즘이 기존의 알고리즘에 사용되는 핑거프린트 용량에서 약 50% 정도를 축소한 용량으로 평균 90%의 매칭율을 보이는 핑거프린트의 성능을 보인다. 또한 매 프레임마다 주파수 분석을 하지 않기 때문에 분석시간 또한 줄어들게 된다.

V. 결론

본 논문에서는 영상 인식에서 사용되는 알고리즘으로 DoG 알고리즘을 이용하여 핑거프린트를 추출하고 DTW 알고리즘을 이용하여 동일한 오디오 데이터를 추출하는 방법을 제시하였다. DoG 알고리즘과 FFT를 통해 오디오 데이터의 핑거프린트 개수를 줄일 수 있었고, 순차적 DTW 검색을 통해 오디오 데이터의 부분 정보를 이용하여 동일 오디오 데이터를 추출 할 수 있었다.

이전의 핑거프린트 추출방식은 모든 프레임에서 주파수 분석을 하였기 때문에 핑거프린트의 데이터양이 많아지고, 속도 또한 느려지게 되었다. 하지만 본 논문에서는 특정 위치 지점에서만 주파수 분석을 하였기 때문에 핑거프린트의 데이터양을 줄이는 동시에 속도를 높일 수 있었다.

본 논문의 연구를 통하여 음악의 특성에 따른 핑거프린트의 위치를 추출 할 수 있었다. 추후 본 논문의 알고리즘을 이용한 음악 매칭을 통해 인터넷에 유통되는 복사된 음악의 저작권 보호, 또는 음악의 메타정보 등을 사용자에게 나타낼 수 있을 것이다.

참고문헌

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Intl. J. Computer Vision, Vol. 60, No. 2, pp. 91-110, 2004.

[2] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," European Conference on Computer Visioin, 2002.

[3] T. Lindeberg, "Scale-space theory: a basic tool for analyzing structures at different scales," Journal of Applied Statistics, 1994.

[4] E. J. Keogh, MJ. Pazzani. "Derivative Dynamic Time Warping," First SIAM international Conference on Data Mining, 2001.

[5] J. C. Brown, A. Hodgins-Davis, PJO. Miller "Classification of vocalizations of killer whales using dynamic time warping," Journal of the Acoustical Sociery of America 119, EL34, 2006.

[6] A. M. Youssef, T.K. Abdel-Galil, E.F. El-Saadany, M.M.A. salama "Disturbance classification utilizing dynamic time warping classifier," IEEE Transactions on Power Delivery, Vol. 19, No. 1, pp. 272-278, 2004.

[7] A. Pikrakis, S. Theodoridis, d. Kamarotos. "Recognition of isolated musical patterns using context dependent dynamic time warping," IEEE. Speech and Audio Processing, Vol. 11, pp. 175-183, 2003.

[8] J. W. Picone, "Signal modeling techniques in speech recognition," Proc. IEEE, Vol. 8, No. 9, pp. 1215-1247, Sept. 1993.

[9] A. Casey, "Content-Based Music Information Retrieval: Current Directions and Futrue Challenges," IEEE, Vol. 96, No. 4, 2008.

[10] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search, and

retrieval of audio," IEEE Multimedia, Vol. 3, No. 2, 1996.
〔11〕 Avery Li-Chun Wang and Julius O. Smith, III., WIPO publication WO 02/11123A2, 7 Feb. 2002.
〔12〕 Shumeet Baluja, Michele Covell, "Waveprint: Efficient Wavelet-Based Audio Fingerprinting," Elsevier Pattern Recognition, 41, 3467-3480, 2008.
〔13〕 C. Jacobs, A. Finkelstein, D. Salesin, "Fast multiresolution image querying," SIGGRAPH 95, 1995.
〔14〕 (주)엔써즈, "*오디오 핑거프린트 데이터 생성 방법 및 장치 및 이를 이용한 오디오 데이터 비교 방법 장치*" 한민국특허, 공개번호 10-2008-0098878.
〔15〕 Lago, N.P., Kon, F, "The Quest for Low Latency," the International Computer Music Conference (ICMC 2004), pp. 33-36. 2004.
〔16〕 "*http://www.independentrecording.net/irn/resources/freq chart/main_display.htm*" IRN, 2006.

저 자 소 개

권 진 만

2008: 숭실대학교 미디어학과 공학사.
2008 - 현재: 숭실대학교 미디어학과 석사과정
관심분야: 패턴인식, HCI, 멀티미디어 정보검색

고 일 주

1992: 숭실대학교 전산학과 공학사
1994: 숭실대학교 전산학과 공학석사
1997: 숭실대학교 전산학과 공학박사
2003-현재: 숭실대학교 미디어학과 조교수
관심분야: 감성인식, 콘텐츠공학, 멀티미디어 정보검색 등

장 대 식

1994: 숭실대학교 전산학과 공학사.
1996: 숭실대학교 전산학과 공학석사.
1999: 숭실대학교 전산학과 공학박사.
2005 - 현재: 군산대 컴퓨터정보공학과 조교수
관심분야: 로봇 비전, 영상 처리