碩 士 學 位 論 文

음악 검색을 위한 오디오 핑거프린팅 방법 및 데이터베이스 검색 알고리즘

Audio Fingerprinting Method and Database Search Algorithm for Music Retrieval

高麗大學校 컴퓨터情報通信大學院

소프트웨어工學 專攻

李 鮮 炯

2011年 8月

陸 東 錫 敎 授 指 導

碩 士 學 位 論 文

음악 검색을 위한 오디오 핑거프린팅 방법 및 데이터베이스 검색 알고리즘

Audio Fingerprinting Method and Database Search Algorithm for Music Retrieval

이 論文을 工學 碩士學位 論文으로 提出함.

2011 年 8 月

高麗大學校 컴퓨터情報通信大學院

소프트웨어工學 專攻

李 鮮 炯

李鮮炯의 工學 碩士學位 論文 審査를 完了함

2011年 8月 日

委員長 육 동 석 (印)

委 員 김 정 현 (印)

委 員 강 재 우 (印)

요 약

본 논문에서는 다양한 음악 관련 서비스를 제공할 수 있는 음악 인식 시스템을 구축하기 위하여 필립스 음악 검색 방법을 기반으로 음악 검색을 위한 최적의 오디오 특징 파라미터 추출 실험을 수행하였다. 실험 결과 입력 신호의 프레임 길이, 간격, 차수 등을 최적화 하였다. 또한 오디오 핑거프린트를 검색하기 위한 lookup table 방식의 성능을 보완하기 위하여 검색대상에 가중치를 부여하는 방법을 제안하였다. 기존의 lookup table 방식은 검색 핑거프린트 블록을 가지고 인덱스에서 매칭되는 노래 전체를 탐색하여 기준 에러율보다 낮고 동시에 가장 낮은 에러율을 가지는 노래를 검색하는 방식이다. 이러한 방식은 곡 수가 증가하고 데이터베이스의 원본 노래와 검색 대상의 노래의 핑거프린트 유사도가 높을수록 검색횟수가 높아지는 단점을 보일 수 있다. 제안된 방법은 인덱스에서 매칭되는 노래를 블록단위로 나누어 가중치를 부여하고, 원본 노래의 핑거프린트와 에러율 비교시 가중치가 높은 노래부터 비교하는 방식이다. 제안된 방식은 lookup table 방법과 유사한 정확도를 나타내었고 검색에 대한 성능이 향상되는 특징을 보였다.

목 차

그 림 목 차

표 목 차

제1장 서론

1.1 연구의 배경

일반적으로 구축할 수 있는 음악 검색 시스템은 음악 파일의 이진 데이터를 기반으로 한 MD5(message-digest algorithm 5)와 같은 해시 값이나 ID3 tag와 같은 음악 파일의 부가 정보를 활용을 하게 된다[1]. 그러나 이러한 정보만 가지고 데이터베이스를 구축하게 되는 경우 음악 제공 업체나 인코더 프로그램의 차이로 인해 해시 값이나 ID3 tag정보가 달라질 수 있어 같은 음악임에도 불구하고 검색이 안되어 데이터베이스에 중복된 데이터를 유지해야 하는 문제가 발생하게 된다.

또한 최근 인터넷의 발달에 따라 디지털 음악 시장이 커지고 있고, 이러한 디지털 음원을 불법 P2P에서의 공유로 인해 세계적으로 저작권 보호 이슈가 야기되고 있다[2]. 이러한 불법적인 공유를 막기 위한 저작권 필터링 기능과 같은 요구사항을 충족하기 위해서는 기존의 파일 기반 검색 시스템으로는 어려움이 존재하게 된다. 따라서 파일 기반이 아닌 오디오 내용을 기반으로 한 음악 검색 시스템이 필요하다

1.2 연구의 목적

본 연구에서는 기존의 오디오 핑거프린트 추출 기법에 대해 분석하여 핑거프린트를 추출 부분에 대해 최적의 파리미터를 도출하고 핑거프린트를 이용하여 원본 음악을 검색하는 look up table 기반의 검색 시스템의 성능을 개선하기 위해 검색 방법에 대한 개선점을 제시하고자 한다.

1.3 연구의 구성

본 연구의 구성은 다음과 같다.

1장에서는 본 연구를 수행하게 된 배경과 목적을 살펴보고 본 연구의 구성을 제시하였다. 2장에서는 현재 일반적으로 사용되고 있는 다양한 오디오 핑거프린트 추출 기법에 대해 설명을 하였다. 3장에서는 본 논문에서 사용한 오디오 핑거프린트 추출 방법에 대한 설명과 검색 성능을 개선하기 위한 방법을 제안한다. 4장에서는 3장에서 제시한 핑거프린트 추출 방법과 검색 성능 개선에 대한 실험을 진행하고 그 결과를 분석하였고, 마지막으로 5장에서는 본 연구에 대한 결론을 제시한다.

제2장 관련연구

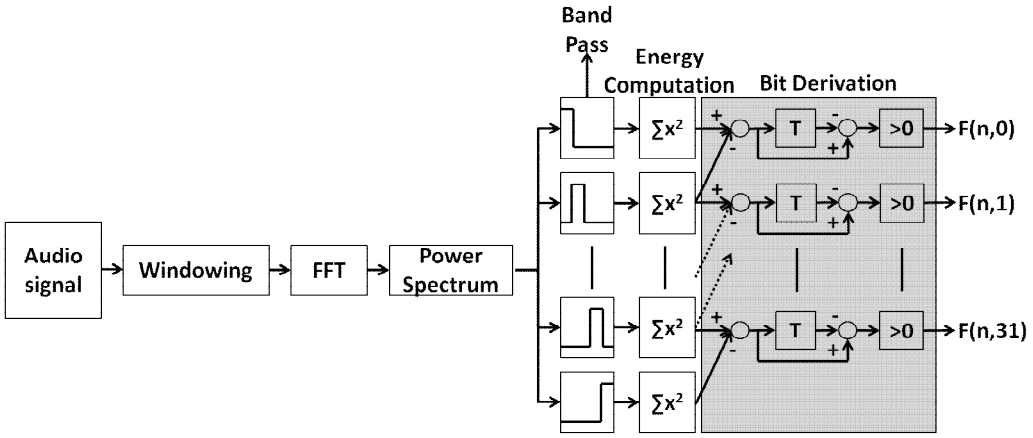
음악 식별 기술은 음악의 신호 특징을 이용하여 임의의 음악을 찾아내는 기술로 흔히 핑거프린팅 기술로 알려져 있다. 핑거프린팅 기술은 오디오 신호에서 인간의 지문에 해당하는 핑거프린트를 추출하여 DB화 하고 이를 기반으로 임의의 음악을 검색하는 기술이다[3]. 핑거프린팅 기술은 일반적으로 다음과 같은 세 가지 조건을 만족하여야 한다[3][4].

- Robustness : 인지적으로 유사한 왜곡에 대해서는 핑거프린트 값도 유사해야 함
- Pairwise Independency : 서로 다른 컨텐츠에서 추출된 핑거프린트 값은 달라야 함
- Search Efficiency : 대용량 핑거프린트에 대해 효율적인 검색이 가능해야 함

이러한 핑거프린팅 기술은 방송 모니터링, 음악 검색 서비스, 저작권 필터링 등으로 널리 활용되고 있으며 핑거프린팅 기술을 보유한 기업으로는 Gracenote, Shazam, Google등이 대표적이며, 이들이 보유한 기술은 다음과 같다[5][6][7].

2.1 필립스 오디오 핑거프린팅 기법

Gracenote에서 사용중인 기술은 필립스에서 제안한 핑거프린팅 기법이며 추출 알고리즘은 그림 2.1과 같다[8].



<그림 2.1> 필립스 오디오 핑거프린트 추출 방법

필립스 오디오 핑거프린트 추출 방법은 5000Hz로 resample되어 입력된 시간 영역의 신호를 일정한 길이의 프레임으로 나누어 FFT(fast Fourier transform)를 이용해 주파수 영역으로 변환하고, 시간-주파수 변화에 따른 에너지 차이를 이진화한 값으로 표현한다. 주파수 영역으로 변환된 신호는 로그 간격을 가지는 300~2000Hz 대역의 중첩되지 않은 33개 주파수 밴드로 나누어 각 밴드의 에너지 크기를 구하고 식(1)과 식(2)를 통해 이진화하게 된다.

$$ED(n,m) = (E(n,m) - E(n,m+1)) - (E(n-1,m) - E(n-1,m+1)) \quad (1)$$

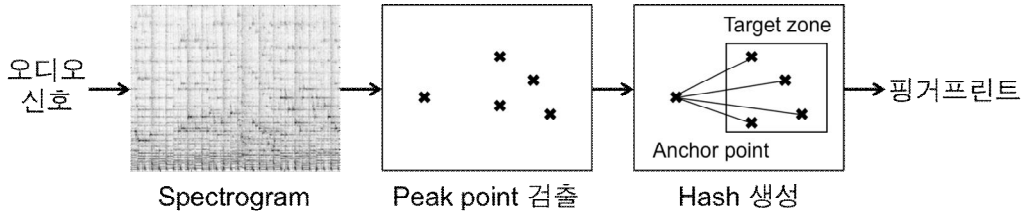
$$F(n,m) = \begin{cases} 1 \ if \ ED(n,m) > 0 \\ 0 \ if \ ED(n,m) \le 0 \end{cases} \quad (2)$$

식(1)에서 E(n,m)은 n번째 프레임의 m번째 주파수 밴드의 에너지 크기를 의미하고, ED(n,m)은 인접한 시간-주파수 변화에 따른 에너지 차이를 의미한다. 이러한 에너지 차이는 식(2)를 통해 이진화되어 32차의 sub-fingerprint로 변환되고 이러한 sub-fingerprint 256개가 모여 하나의 핑거프린트 블록을 구성하게 된다. 핑거프린트의 유사도는 검색 대상과 원본 핑거프린트 블록간의 Hamming distance로 bit error를 계산하고 BER(Bit Error Rate)로 표현한다. 검색 대상 핑거프린트의 블록의 길이를 F라고 할 때 BER은 식(3)과 같다.

$$BER = HammDist\ (Original\ , Target\ ) / F \quad (3)$$

2.2 Shazam 핑거프린팅 기법

Shazam에서 보유하고 있는 핑거프린팅 알고리즘은 그림 2.3과 같다[9].

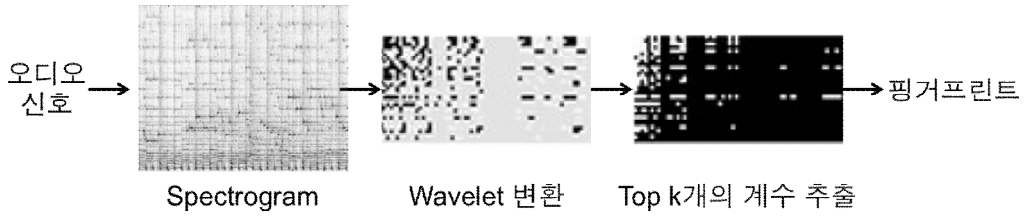


<그림 2.2> Shazam 오디오 핑거프린트 추출 방법

STFT(short time Fourier transform)을 이용하여 입력된 오디오 신호를 시간 영역에서 주파수 영역으로 변환을 한 다음 매 프레임 마다 frequency bin에서 peak point를 검출한다. 이것을 constellation map이라 부르고, sparse set을 생성하여 특징을 추출하고 음악을 검색하게 된다. 비슷한 노래는 시간차와 주파수차가 비슷할 것이라는 가정을 기반으로 해시 값을 생성하게 된다. 하나의 해시 값은 anchor point에 대해 target zone에 대한 시간 및 주파수 차이 정보를 이용하여 구성하며 이를 핑거프린트로 사용한다[9].

2.3 구글 오디오 핑거프린팅 기법

Google이 개발한 기술은 오디오 신호에 이미지 처리 기법을 도입하여 특징을 추출하며 그림 2.4와 같다[10].



<그림 2.3> 구글 오디오 핑거프린트 추출 방법

시간 영역의 오디오 신호를 STFT(short time Fourier Transform)을 이용하여 주파수 영역으로 변환하여 스펙트럼 이미지를 얻은 후, 이 이미지에 웨이블릿(wavelet) 변환을 하고 Top k개의 웨이블릿 계수 값을 이용하여 이진화 한 다음 이를 핑거프린트로 이용한다[10].

제3장 오디오 핑거프린트 추출 방법 및 검색 알고리즘

3.1 오디오 핑거프린트 추출 방법

본 논문에서는 필립스의 오디오 핑거프린트 추출 방법을 기반으로 인접한 주파수 밴드간 중첩도가 50%이고 triangle filter를 적용하여 주파수 밴드의 파워 스펙트럼을 추출하였고 다음과 같은 특징 추출 파라미터를 최적화하기 위한 실험을 진행하였다

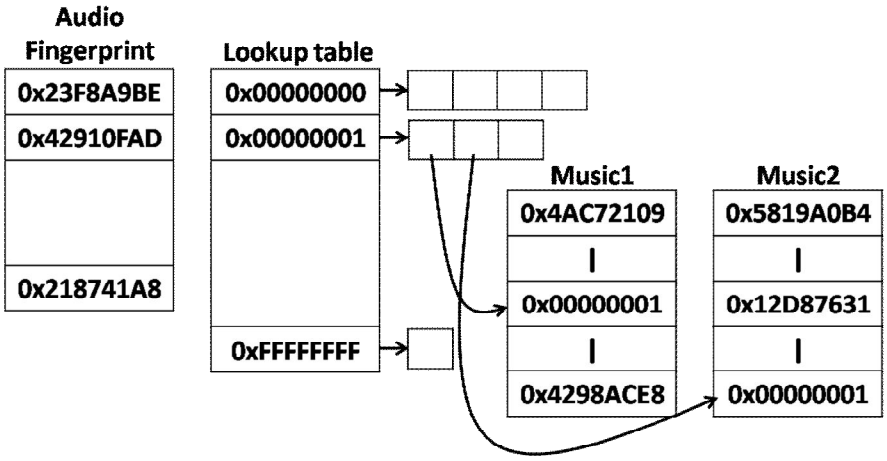
- 특징 추출 방식
- 입력 신호의 주파수 범위
- Bark scale을 가지는 주파수 밴드의 차수
- 프레임 길이
- 프레임 간격

추출된 핑거프린트의 BER과 음악 검색 성능을 기반으로 실험 결과를 분석하였다.

3.2 검색 알고리즘

3.2.1 Lookup table 기반 인덱스 검색 알고리즘

그림 3.1은 기존의 lookup table 기반의 오디오 핑거프린트 검색 데이터베이스 구조를 나타낸다[8].



<그림 3.1> lookup-table 데이터베이스 레이아웃

Lookup table은 sub-fingerprint 값을 key로 하고 원곡에서 sub-fingerprint 위치 정보를 value로 하여 데이터베이스의 인덱스를 구성하게 된다. 검색하고자 하는 샘플 음악의 sub-fingerprint 값과 동일한 sub-fingerprint를 가지는 음악의 위치를 lookup table에서 찾은 뒤

데이터베이스에서 그 위치의 오디오 핑거프린트와 샘플 음악의 오디오 핑거프린트 사이의 BER을 계산한다. BER이 최소인 음악의 BER이 사전에 정해진 threshold보다 낮으면 검색이 성공적으로 완료한다. 필립스 오디오 핑거프린트 추출 방법에서는 threshold BER값을 0.35로 제시하고 있다[8]. Lookup table 기반의 검색 알고리즘에 대한 의사 코드는 표 3.1과 같다.

[표 3.1] lookup table 기반 검색 알고리즘의 의사코드

```
queryBlock = fingerprint block for find
bestMusic = none
minimumBER = 1.0

for i from 0 to queryBlock.size - 1
begin
  indexNode = LUTIndex.find(queryBlock[i])

  for j from 0 to indexNode.child_count - 1
  begin
    ber = CalcuateBitError(indexNode, queryBlock)
    IF ber <= 0.35 AND ber < minimumBER
    begin
      bestMusic = indexNode.music_id
      minimumBER = ber
    end

    indexNode = indexNode.Next
  end
end

return bestMusic
```

Lookup table기반의 검색 알고리즘은 검색하고자 하는 샘플 음악과 원본 노래의 해당 구간에 같은 값을 가지는 sub-fingerprint가 적어도 1개

존재한다는 가정하에 검색을 하게 되는데, 입력 샘플의 왜곡이 심한 경우에는 동일한 값을 가지는 sub-fingerprint가 존재하지 않을 수 있다. 이러한 경우에는 sub-fingerprint에서 임의의 1비트를 반전하여 인덱스를 재 탐색하게 된다. 일정 횟수 이상 재 탐색을 하여도 찾을 수 없는 경우 데이터베이스에 샘플 음악이 없다고 판단하고 종료하게 된다.

32차 sub-fingerprint를 사용하는 경우 데이터베이스에 10,000곡이 있을 때 전체 sub-fingerprint개수는 약 2억 5,000만개 정도가 되는데 lookup table 기반으로 데이터베이스 검색을 하게 되면 데이터베이스의 오디오 핑거프린트 조회 횟수는 query 당 평균 약 300회가 된다[8]. 이것은 데이터베이스에 있는 음악의 개수에 비례하므로, 대용량 음악 검색의 경우 많은 데이터베이스 조회, 즉 디스크 I/O를 초래한다.

또한 검색하고자 하는 샘플 음악이 잡음 등에 의해서 왜곡된 경우, sub-fingerprint를 lookup table에서 찾지 못할 수 있기 때문에 데이터베이스 조회 횟수는 더욱 늘어날 수 있다. 그리고 sub-fingerprint 차수에 따라 잡음에 대한 민감도가 달라지게 되는데 이러한 민감도를 줄이기 위해서 차수를 낮춘다면 이는 곧 데이터베이스 조회 횟수의 증가로 인해 검색 성능의 하락으로 이어지게 된다[11]. 이러한 단점을 개선하고자 본 논문에서는 잡음에 강인한 대용량 음악을 위해서 낮은 차수의 sub-fingerprint를 사용하고 동시에 데이터베이스 조회 수를 줄일 수 있는 효율적인 오디오 핑거프린트 검색 알고리즘을 제안한다.

3.2.2 개선된 검색 알고리즘 제안

32차 오디오 특징을 사용하게 되는 경우 평균 검색 횟수는 적지만 주파수 밴드가 세분화되어 오디오 신호에 노이즈가 포함된 경우 검색 정확도가 떨어지게 된다[11]. 이러한 문제를 보완하고자 오디오 특징의 차수를 줄이게 되면 노이즈에 강해질 수는 있지만 sub-fingerprint의 표현 가능한 수가 적어져 인덱스 상의 각 sub-fingerprint가 가리키는 음악이 많아지게 되고, 결국 평균 검색 횟수가 증가하는 문제가 발생하게 된다. 이렇게 검색 횟수가 증가하는 문제를 개선하고자 본 논문에서는 노래를 블록 단위로 나누어 우선순위를 부여하는 방법을 제안한다.

1) Lookup table에서 검색 후보 음악 추출 : 검색하고자 하는 샘플 음악의 sub-fingerprint들이 발현된 음악들을 lookup table에서 탐색하여 각 음악 별로 sub-fingerprint가 발현된 위치 순서대로 정렬을 한다.

2) 검색 후보 음악에 가중치 부여 : 검색 우선 순위를 부여하기 위한 가중치를 계산하기 위해 샘플 음악의 sub-fingerprint들이 원본 노래에서 발현된 횟수와 발현된 위치에 대한 표준편차를 사용한다. 우선, sub-fingerprint가 발현된 음악을 블록으로 나눈다. 우선순위는

곡 단위로 하지 않고 블록 단위로 계산한다. 예를 들면 노래에서 반복이 많이 되는 후렴구의 오디오 신호로 음악을 검색하는 경우 노래 안에서 2~3곳 이상 유사한 블록이 발견될 수 있다. 또한 검색 대상과 전혀 다른 노래인 경우에도 해당 노래의 전체에 걸쳐 같은 값을 가지는 sub-fingerprint가 산발적으로 분포할 수 있다. 이러한 경우 곡 단위로 가중치를 계산하게 되면 같은 값을 가지는 sub-fingerprint의 분포의 표준편차가 커지기 때문에 동일한 노래임에도 불구하고 우선순위가 하락할 수 있게 된다. 후보 블록의 크기는 한 개의 후보 음악 전체에서 검색 샘플의 sub-fingerprint가 고유하게 발현한 횟수를 사용한다. 후보 블록 안에서 검색 대상 sub-fingerprint가 고유하게 발생하는 횟수를 구하여 UV라 하고, sub-fingerprint가 발현한 위치에 대한 표준편차를 S라고 할 때 후보 블록의 가중치 P는 식(4)로 구할 수 있다.

$$P = UV / S \tag{4}$$

가중치를 식(4)와 같이 구하는 이유는 블록 안에서 중복되지 않은 sub-fingerprint가 많이 발현되고, sub-fingerprint가 발현된 위치의 간격이 좁을수록 유사도가 높기 때문이다. 이렇게 계산된 가중치는 값이 클수록 우선순위가 높은 블록이 된다.

3) 가중치 순서로 reordering하여 데이터베이스 탐색 : 가중치 순서로 정렬된 후보 블록을 가지고 lookup table 인덱스를 탐색하여 원본 노래의 핑거프린트 블록과 검색 핑거프린트 블록의 BER을 계산하여 threshold BER보다 낮은 음악을 찾게 되면 해당 음악을 결과로서 반환한다. Threshold BER보다 낮은 음악을 찾기 못하면 기존 검색 방식과 동일하게 샘플 핑거프린트의 sub-fingerprint에서 임의의 1비트를 반전하여 검색을 재시도한다.

표 3.2는 제안된 알고리즘을 의사코드로 표현한 것이다.

[표 3.2] 개선된 검색 알고리즘의 의사코드

```
queryBlock = fingerprint block for find
targetMusic = music list for compute candidate block

//Extract target music from lookup table index
for i from 0 to queryBlock.size - 1
begin
  indexNode = LUTIndex.find(queryBlock[i])

  for j from 0 to indexNode.child_count - 1
  begin
    targetMusic.add(indexNode)
    indexNode = indexNode.Next
  end
end

//Calculate priority of candidate blocks from target music
candidateBlocks = none
for i from 0 to targetMusic.music_count - 1
begin
  currentMusic = targetMusic[i]
  uniqueSFP = GetUniqueSFPCount(currentMusic)
```


```
  for j from 0 to currentMusic.position_count - uniqueSFP
  begin
    [blockStDev, blockUniqueSFP] = CalcStDevAndUniqueCount(j, uniqueSFP)
    candidateBlocks.add(blockUniqueSFP / blockStDev,
                        MakeBlock(i, uniqueSFP))
  end
end

//Search
bestMusic = none

isFindMusic = false
for i from 0 to queryBlock.size - 1
begin
  indexNode = LUTIndex.find(queryBlock[i])

  for j from 0 to indexNode.child_count - 1
  begin
    ber = CalcuateBitError(indexNode, queryBlock)
    IF ber <= 0.35
    begin
      bestMusic = indexNode.music_id
      isFindMusic = true
      break
    end

    if(isFindMusic == true) break
    indexNode = indexNode.Next
  end
end

return bestMusic
```


제4장 실험 결과

4.1 오디오 핑거프린트 추출 실험

4.1.1 실험 데이터

오디오 핑거프린트 추출 실험을 위한 데이터는 다양한 장르의 MP3 파일 1,000개로 구성하였다. 검색 실험에 사용할 음악 샘플은 원본 MP3 파일 중에서 임의로 약 3.1초의 구간을 32kbps와 128kbps로 변환하여 사용하였다. 평균 BER을 측정하기 위하여 찾고자하는 음악이 원본에 존재하는 positive 샘플의 경우 원본 MP3와 비교하였고, 찾고자하는 음악이 원본에 없는 negative 샘플 실험의 경우 1,000곡에서 자신을 제외한 임의의 10곡과 비교하였다. 평균 BER의 품질을 평가하기 위하여 positive 샘플과 negative 샘플의 BER 분포에 대한 PDF(probability density function)를 추정하고 PDF사이의 거리를 측정하였다. Positive 샘플과 negative 샘플의 BER을 각각 BER_P, BER_N 이라고 하고 각각의 분산을 V_P, V_N이라 할 때 PDF 거리는 식(5)로 계산할 수 있다.

$$PDF\ Dist = (\mathrm{BER_N} - \mathrm{BER_P})^2 \ / \ \sqrt{V_N \times V_P} \qquad (5)$$

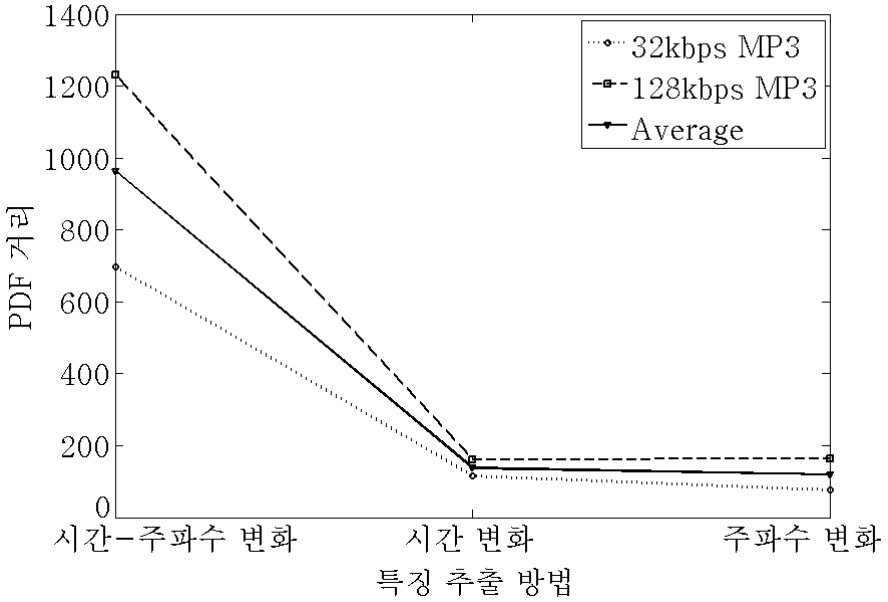
검색 정확도를 측정하기 위하여 positive 샘플의 경우 1,000개의 원본 MP3 음악을 대상으로 검색하였고, negative 샘플의 경우 찾고자 하는 노래를 제외한 999개의 원본 MP3 음악을 대상으로 검색하였다. 검색 실험에 사용한 샘플에는 원본 MP3 파일을 32kbps, 128kbps로 변환한 샘플과 128kbps 음악에 SNR(sound to noise ratio)이 30dB이 되도록 백색 잡음을 추가한 샘플을 사용하였다.

4.1.2 특징 추출 방식 실험

특징 추출 방식 실험은 시간-주파수의 변화, 시간 변화, 주파수 변화 등 세 가지 방식 별로 주파수 밴드의 에너지 차이를 이진화하여 핑거프린트를 추출하였다.

[표 4.1] 특징 추출 방법별 평균 BER

		시간-주파수 변화	시간 변화	주파수 변화
Positive 샘플	32kbps MP3	0.125	0.118	0.087
	128kbps MP3	0.078	0.079	0.048
Negative 샘플	32kbps MP3	0.446	0.418	0.391
	128kbps MP3	0.446	0.416	0.386



<그림 4.1> 특징 추출 방법별 BER 분포의 PDF 거리

표 4.1에서 positive 샘플의 평균 BER 값은 주파수 변화에 대한 에너지 차이를 특징으로 한 결과가 가장 좋은 것으로 보이나 그림 4.1과 같이 BER 분포에 대한 PDF 거리를 비교해 보면 시간-주파수 변화에 따른 에너지 차이를 특징 추출로 사용하는 것이 가장 성능이 좋은 것을 확인하였다.

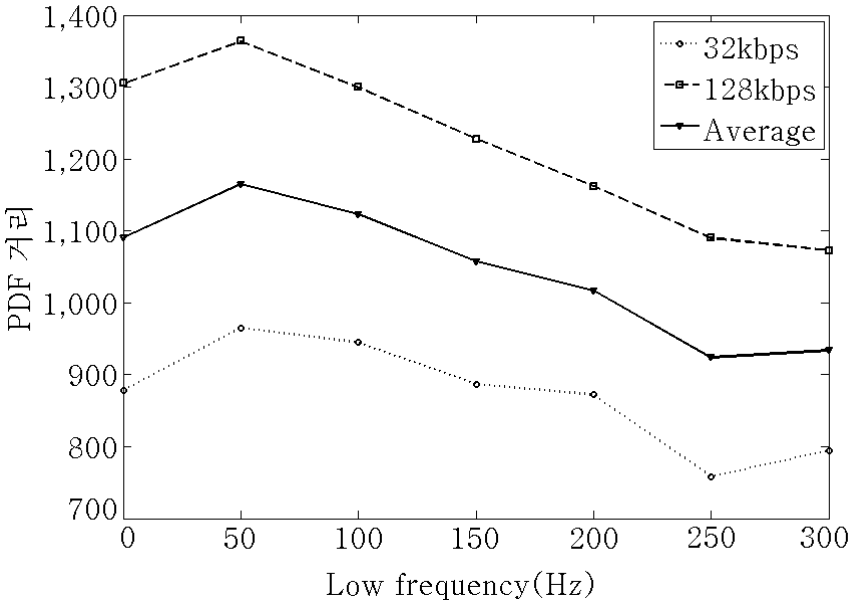
4.1.3 주파수 범위 실험

입력 신호의 주파수 범위를 결정하는 실험은 low frequency cut-off를 0~300Hz까지 50Hz씩 증가하면서 수행하였다. High frequency cut-off는 오디오 신호가 5000Hz로 리샘플 되었기때문에 Nyquist frequency인 2500Hz를 사용하였고 특징 추출 방식은 시간-주파수 변화에 따른 에너지 차이를 이진화하는 방법을 사용하였다.

표 4.2의 평균 BER 결과를 보면 low frequency가 증가할수록 positive 샘플의 평균 BER이 증가하고 negative 샘플의 평균 BER은 큰 변화가 없는 것을 볼 수 있지만 각 low frequency의 BER 분포에 대한 PDF 거리를 비교한 그림 4.2를 보면 50Hz에서 positive와 negative 샘플의 BER 분포가 가장 멀게 나타나 성능이 좋은 것을 확인하였다.

[표 4.2] Low frequency별 평균 BER

		0Hz	50Hz	100Hz	150Hz	200Hz	250Hz	300Hz
Positive 샘플	32kbps MP3	0.107	0.107	0.110	0.115	0.120	0.125	0.129
	128kbps MP3	0.060	0.061	0.063	0.066	0.069	0.072	0.076
Negative 샘플	32kbps MP3	0.444	0.445	0.445	0.445	0.445	0.444	0.445
	128kbps MP3	0.444	0.445	0.445	0.445	0.445	0.444	0.445



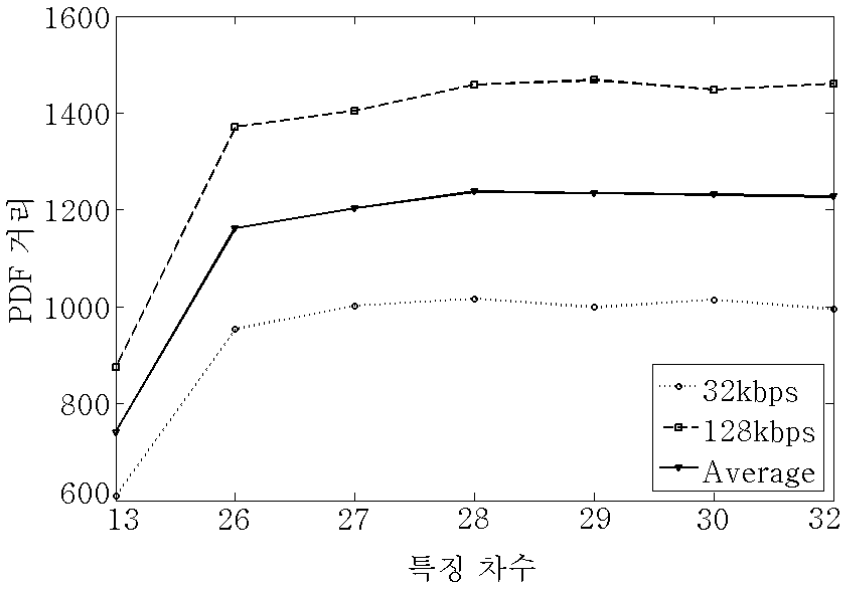
<그림 4.2> Low frequency별 BER 분포의 PDF 거리

4.1.4 특징 차수 실험

특징의 최적 차수를 결정하는 실험은 차수를 13~32차로 변화하며 진행하였다. 특징 추출 방법은 시간-주파수 변화에 따른 에너지 차이를 이진화 하는 방법을 사용하였고 입력 주파수의 범위는 50~2500Hz이다. 주요 차수의 평균 BER 결과는 표 4.3과 같다. 특징 차수별 평균 BER 값과 그림 4.3의 BER 분포의 PDF 거리를 분석한 결과 28차 특징에서 가장 좋은 성능을 나타내었다.

[표 4.3] 특징 차수별 평균 BER

		13차	26차	27차	28차	29차	30차	32차
Positive 샘플	32kbps MP3	0.085	0.092	0.092	0.092	0.093	0.093	0.093
	128kbps MP3	0.047	0.051	0.051	0.051	0.052	0.052	0.052
Negative 샘플	32kbps MP3	0.412	0.440	0.441	0.442	0.442	0.443	0.444
	128kbps MP3	0.412	0.440	0.441	0.442	0.442	0.443	0.444



<그림 4.3> 주요 특징 차수별 BER 분포의 PDF 거리

28차 특징을 이용한 검색 실험 결과는 표 4.4와 같다. 13차 특징은 주파수 밴드를 Bark scale의 critical band로 나누어 특징을 추출하였고, 32차 특징은 필립스 방식에서 제안한 특징 차수이다. 낮은 차수를 사용할수록 잡음에 강인한 검색 성능을 보이고, 높은 차수를 사용할수록 검색 속도가 향상되는 것을 관측하였다. 이것은 높은 차수의 핑거프린트 일수록 중복될 확률이 낮기 때문일 것이다.

[표 4.4] 특징 차수별 검색 정확도(%) 및 평균 탐색 횟수

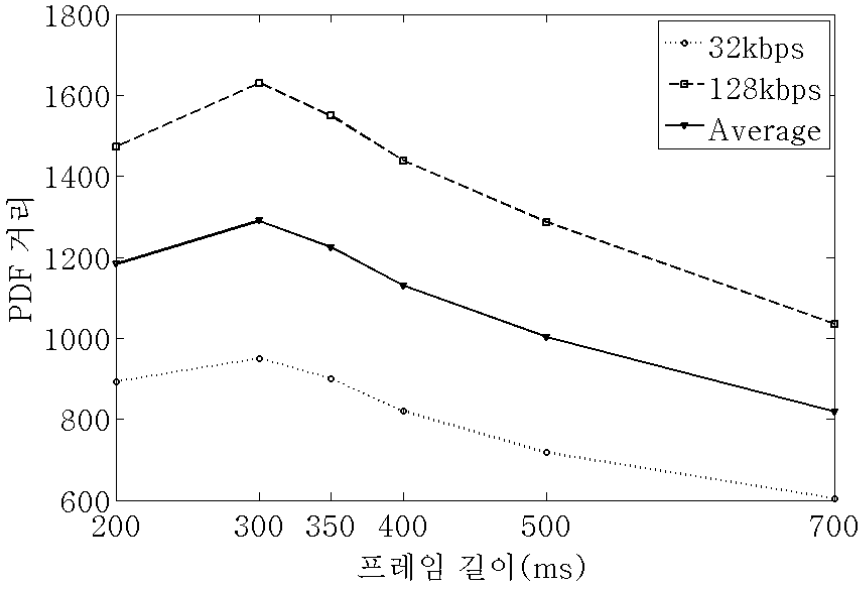
	Positive 정확도	Negative 정확도	Positive 탐색 횟수	Negative 탐색 횟수
13차	99.9	99.6	1.8E06	3.0E06
28차	99.6	99.9	319	334
32차	99.3	99.9	124	74

4.1.5 프레임 길이 실험

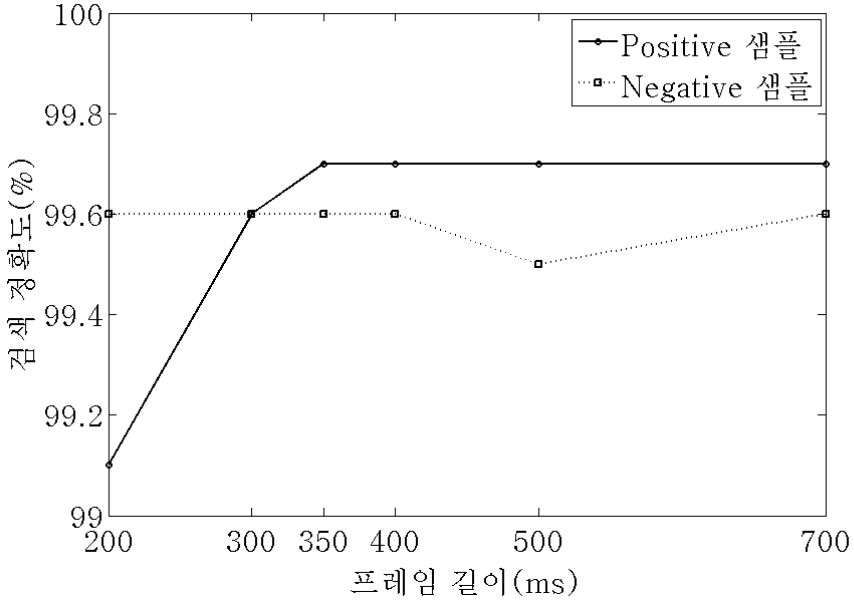
프레임 길이 실험은 프레임 간격을 10ms으로 고정하고 프레임 길이를 200~700ms으로 변화하며 실험을 진행하였다. 특징 추출 방법은 시간-주파수 변화에 따른 에너지 차이를 이진화 하는 방법을 사용하였고 입력 주파수의 범위는 50~2500Hz, 특징의 차수는 28차를 사용하였다. 실험 결과 표 4.5의 평균 BER은 positive/negative 샘플에 관계없이 프레임 길이가 길어질수록 낮아지는 것을 확인할 수 있다. 그러나 그림 4.4의 PDF 거리를 비교해 보면 300ms에서 최고값을 보이다가 감소하는 것을 확인할 수 있다. 그림 4.5의 검색 정확도 결과를 보면 프레임 길이가 350ms와 400ms인 경우에 가장 높은 검색 정확도를 보이고, 그림 4.6의 평균 검색 횟수 결과에서는 프레임 길이가 길어질 수록 검색 횟수가 증가하기 때문에 검색 정확도와 검색 횟수를 고려했을 때 350ms에서 가장 좋은 성능을 나타냄을 확인하였다.

[표 4.5] 프레임 길이별 평균 BER

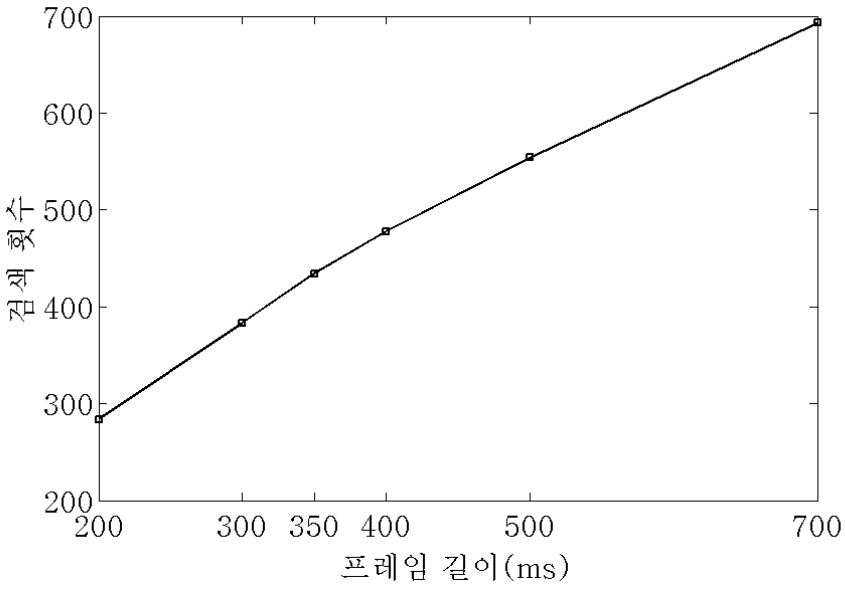
		200ms	300ms	350ms	400ms	500ms	700ms
Positive 샘플	32kbps MP3	0.130	0.108	0.102	0.096	0.087	0.076
	128kbps MP3	0.086	0.069	0.064	0.060	0.053	0.046
Negative 샘플	32kbps MP3	0.452	0.448	0.445	0.441	0.435	0.425
	128kbps MP3	0.452	0.448	0.444	0.441	0.435	0.425



<그림 4.4> 프레임 길이별 BER 분포의 PDF 거리



<그림 4.5> 프레임 길이에 따른 검색 정확도



<그림 4.6> 프레임 길이에 따른 평균 검색 횟수

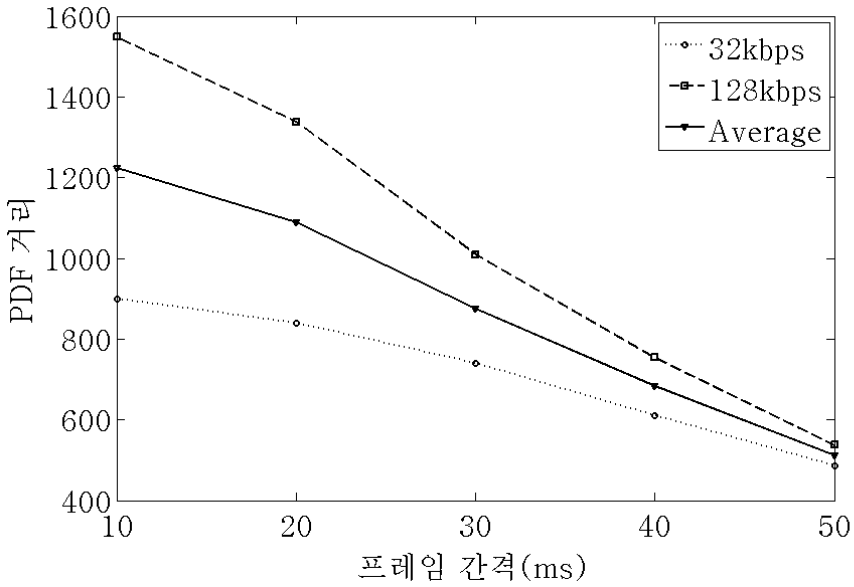
4.1.6 프레임 간격 실험

프레임 간격에 대한 실험은 프레임 길이를 350ms으로 고정하고 프레임 간격을 10~50ms 으로 변화하며 실험을 진행하였다. 특징 추출 방법은 시간-주파수 변화에 따른 에너지 차이를 이진화 하는 방법을 사용하였고 입력 주파수의 범위는 50~2500Hz, 특징의 차수는 28차를 사용하였다. BER 비교 및 검색 실험 시 동일한 길이의 오디오 신호인 약 3초를 사용하였다.

프레임 간격 실험 결과 표 4.6의 평균 BER 값은 프레임 간격이 길어질수록 증가하는 경향을 보이고 그림 4.7의 BER 분포의 PDF 거리 측정 결과에서는 프레임 간격이 10ms에서 가장 성능이 좋고 프레임 간격이 길어질 수록 PDF 거리가 짧아짐을 확인하였다.

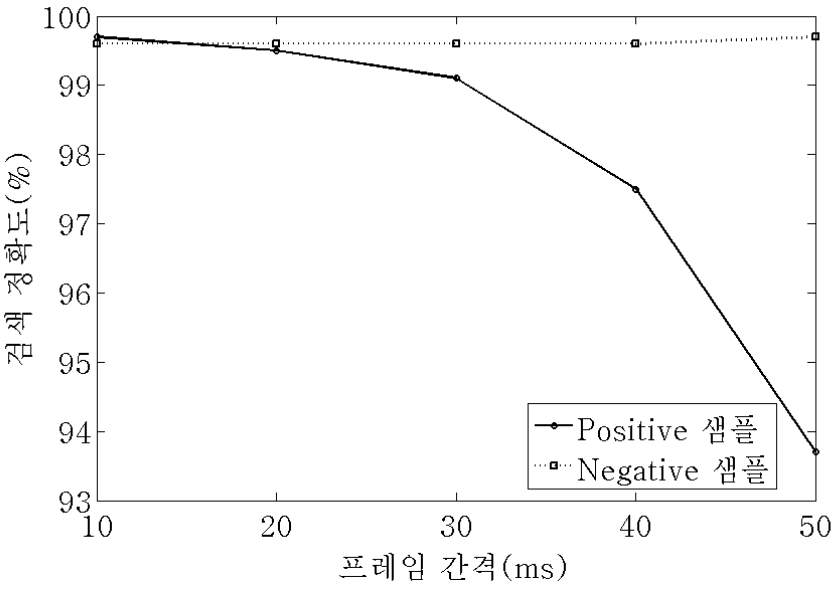
[표 4.6] 프레임 간격별 평균 BER

		10ms	20ms	30ms	40ms	50ms
Positive 샘플	32kbps MP3	0.102	0.107	0.114	0.121	0.129
	128kbps MP3	0.064	0.072	0.080	0.090	0.101
Negative 샘플	32kbps MP3	0.445	0.444	0.444	0.443	0.442
	128kbps MP3	0.444	0.444	0.444	0.443	0.442

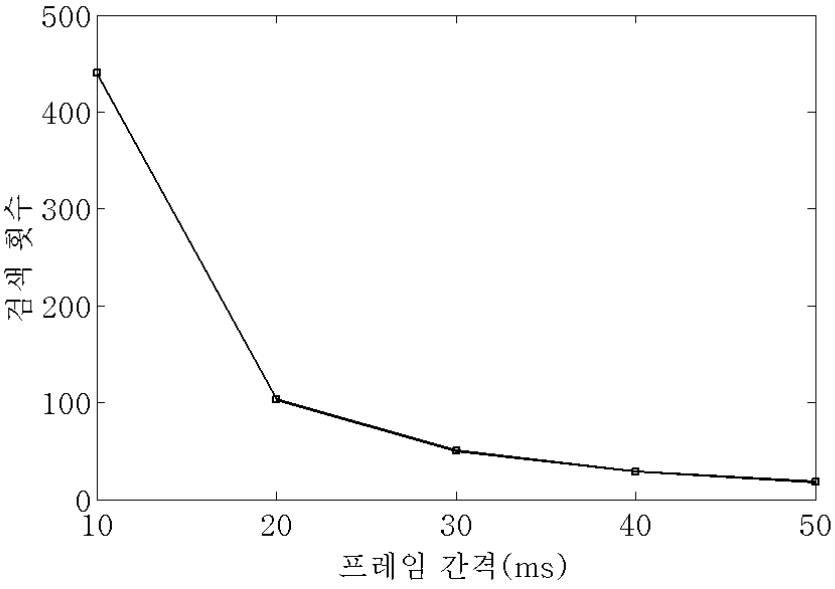


<그림 4.7> 프레임 간격별 BER 분포의 PDF 거리

프레임 간격별 검색 실험에서 검색 정확도는 그림 4.8과 같이 프레임 간격이 짧을수록 정확도가 높은 것을 볼 수 있다. 반면에 평균 검색 횟수는 프레임 간격이 길수록 낮아지는데 이는 동일한 시간의 오디오 신호를 사용하여 검색 실험을 진행하였기 때문에 프레임 간격이 10ms인 경우가 검색 핑거프린트 블록의 길이가 가장 길고 50ms에서 가장 짧기 때문이다.



<그림 4.8> 프레임 간격에 따른 검색 정확도



<그림 4.9> 프레임 간격에 따른 평균 검색 횟수

4.2 검색 알고리즘 실험

4.2.1 실험 데이터

음악 검색을 위한 데이터베이스는 192~320Kbps의 고음질 MP3 음악 1,000개로 구성되었으며 검색 샘플은 128Kbps 음악에 SNR이 각각 5dB, 10dB, 15dB, 20dB, 25dB, 30dB이 되도록 백색 잡음을 추가한 MP3 음악 1,000개로 구성되었다. 검색 실험을 진행하기 위하여 positive 샘플 실험의 경우 1,000개의 원본 MP3 음악을 대상으로 검색하였고, negative 샘플의 경우 찾고자 하는 음악을 제외한 나머지 999개의 원본 MP3 음악을 대상으로 검색하였다. 검색 핑거프린트 블록의 크기는 필립스 핑거프린트 방법과 유사한 약 3초에 해당하는 길이를 사용하였다.

4.2.2 실험 결과

실험은 lookup table 인덱스를 이용하여 검색 샘플 핑거프린트 블록 전체를 탐색하는 “전체 검색”과 검색 중 threshold BER 이내의 노래가 나오면 검색을 종료하는 “최초 검색”, 그리고 제안된 “후보 블록 검색” 방식 세 가지에 대해 진행하였다.

표 4.7과 같이 평균 검색 정확도는 negative 샘플에 대해서는 모든 케이스에서 비슷하게 나왔지만 positive 샘플에서는 “전체 검색”과 “후보 블록 검색”이 유사하게 나왔고 “최초 검색”의 경우에는 다른 두 방식에 비해 낮은 정확도가 나옴을 확인할 수 있다.

[표 4.7] 검색 방법별 평균 검색 정확도

		전체 검색	최초 검색	후보 블록 검색
Positive 샘플	SNR 5dB	41.8%	39%	41.6%
	SNR 10dB	70.2%	68.5%	70.1%
	SNR 15dB	85.6%	85.1%	85.5%
	SNR 20dB	92.5%	92.1%	92.5%
	SNR 25dB	96.8%	96.6%	96.8%
	SNR 30dB	99.2%	99.1%	99%
	평균	81%	80.1%	80.9%
Negative 샘플	SNR 5dB	100%	100%	100%
	SNR 10dB	99.9%	100%	100%
	SNR 15dB	99.7%	99.8%	99.8%
	SNR 20dB	99.8%	99.8%	99.8%
	SNR 25dB	99.7%	99.7%	99.7%
	SNR 30dB	99.6%	99.6%	99.6%
	평균	99.8%	99.8%	99.8%

[표 4.8] 검색 방법별 평균 검색 횟수 및 평균 인덱스 접근 횟수

		전체 검색	최초 검색	후보 블록 검색	
				검색 횟수	인덱스 접근 횟수
Positive 샘플	SNR 5dB	197	146	130	203
	SNR 10dB	255	90	93	500
	SNR 15dB	280	53	80	1391
	SNR 20dB	332	32	76	2881
	SNR 25dB	378	19	79	4675
	SNR 30dB	428	9	81	7179
	평균	307	58	90	2805
Negative 샘플	SNR 5dB	226	226	192	177
	SNR 10dB	316	316	295	259
	SNR 15dB	370	370	349	339
	SNR 20dB	382	382	361	390
	SNR 25dB	406	405	432	432
	SNR 30dB	457	456	454	468
	평균	360	359	347	344

표 4.8과 같이 검색 핑거프린트 블록과 원본 음악의 유사도를 비교한 “검색 횟수”는 positive 샘플의 경우 “전체 검색”이 가장 많은 탐색 횟수를 나타냈고 “후보 블록 검색”이 그 다음, “최초 검색”이 가장 적은 탐색 횟수를 나타냄을 확인하였고 negative 샘플의 경우에는 세 방식이 비슷한 검색 횟수를 나타내었다. 검색을 위한 데이터베이스는 1,000곡의 음악으로

이루어져 있는데 음악의 평균 길이를 5분이라고 할 때 약 3,000만개의 sub-fingerprint가 존재하게 된다. 이 sub-fingerprint가 인덱스 안에 균등하게 분포하고 있다고 하면 인덱스에 sub-fingerprint가 평균적으로 위치하는 개수는 약 0.1(= $30 \cdot 10^6 / 2^{28}$)이 된다. 이러한 경우 하나의 sub-fingerprint가 여러 곡에 중복되어 나타날 확률도 매우 낮아지기 때문에 "최초 검색"의 경우 적은 검색 횟수로 대상 음악을 찾을 수 있게 된다. 그러나 SNR 5dB 샘플과 같이 노이즈가 많은 경우에는 추출된 핑거프린트 블록에서 원본 음악과 동일한 값을 가지는 sub-fingerprint가 적어지게 되므로 높은 검색 횟수가 나타나게 된다

“후보 블록 검색”과 “전체 검색”의 성능 차이를 좀 더 자세히 비교하기 위해서는 “후보 블록 검색”의 “인덱스 접근 횟수”를 “검색 횟수”로 환산할 필요가 있다. “검색 횟수”은 원본 음악과 검색 핑거프린트 블록의 BER을 계산하는 횟수이고, “인덱스 접근 횟수”는 후보 블록의 가중치를 계산하기 위해 lookup table 인덱스가 적재된 메모리에서 데이터를 읽는 횟수를 의미한다. 원본 음악과 검색 핑거프린트 블록간의 BER 계산은 sub-fingerprint 별로 Hamming distance를 계산하여 누적한 값을 핑거프린트 블록간의 BER로 사용하기 때문에 검색 핑거프린트 블록의 크기 F만큼 원본 음악의 핑거프린트가 적재된 메모리를 접근하여 데이터를 읽어오게 된다. 이러한 관계를 이용하여 “인덱스 접근 횟수” I를 “검색 횟수”와 메모리 접근 횟수 관점에서 동일한 단위로 환산한 S는 식(6)과 같다.

$$S = I / F \quad (6)$$

식(6)을 이용하여 인덱스 접근 횟수가 가장 많은 SNR 30dB의 positive 샘플에 대해 환산을 해보면 인덱스 접근 횟수 I = 7179 이고 검색 핑거프린트 블록의 크기는 310개로 하였으므로 F = 310 이므로 환산된 검색 횟수 S는 약 23회가 된다. 이 S를 검색 횟수인 81과 합하면 “전체 검색”의 “검색 횟수”와 동일한 단위가 된다. 이것을 “전체 검색” 결과와 비교해 보면 평균 검색 횟수가 428회인 “전체 검색” 보다 “후보 블록 검색”이 약 4배 정도 검색 속도에서 성능 향상이 있는 것을 볼 수 있다. 검색 정확도 면에서도 속도가 가장 빠른 “최초 검색”보다 높고, 정확도가 제일 높은 “전체 검색”과 비슷한 것을 확인하였다. 검색 결과 전체에 대해 환산을 해보면 positive 샘플에 대해서는 “전체 검색”의 평균 검색 횟수에 비해 약 1.5 ~ 4배 정도의 성능 향상을 보이고 노이즈가 적은 샘플일 수록 성능 향상의 폭이 커지는 것을 확인할 수 있었다. 원본 노래에 대한 핑거프린트를 메모리에 적재하지 않고 디스크에서 직접 읽는 방식으로 구성되어 있다고 가정할 때의 성능 향상을 계산해보면 다음과 같다. SATA3 방식의 하드 디스크와 DDR3 SDRAM의 PC3-10600 모듈의 specification상의 최대 전송 속도가 각각 약 600MB/s와 10,666MB/s로 DDR3 SDRAM이 하드 디스크에 비해 약 18배 정도 속도가 빠르다[12][13]. 이러한 속도 차이를 VD라 할 때 가중치 계산을 위해 메모리에 접근하는

횟수를 하드 디스크에 접근하는 횟수로 환산을 한 HA는 식(7)과 같다.

$$HA = S / VD \quad (7)$$

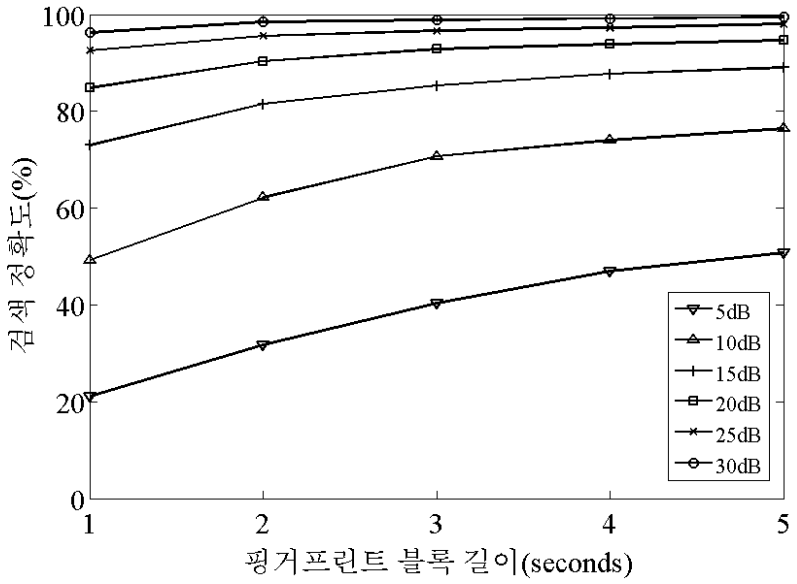
식(6)으로 환산된 SNR 30dB의 positive 샘플의 S를 식(7)로 재 환산하면 HA는 약 1.2가 되고 이는 가중치 설정을 위한 작업이 원본 노래의 핑거프린트 블록 두 개를 하드 디스크에서 직접 읽는 것 보다 빠르다는 것을 알 수 있다.

Positive 샘플의 true positive 결과에 대해 후보 블록 리스트 내 위치와 전체 후보 블록 수 결과인 표 4.9를 보면 평균 후보 블록의 수가 300개 이내이고 후보 리스트에서 평균적으로 상위 20%이내에 검색 대상 노래의 후보 블록이 존재하는 것을 확인할 수 있다.

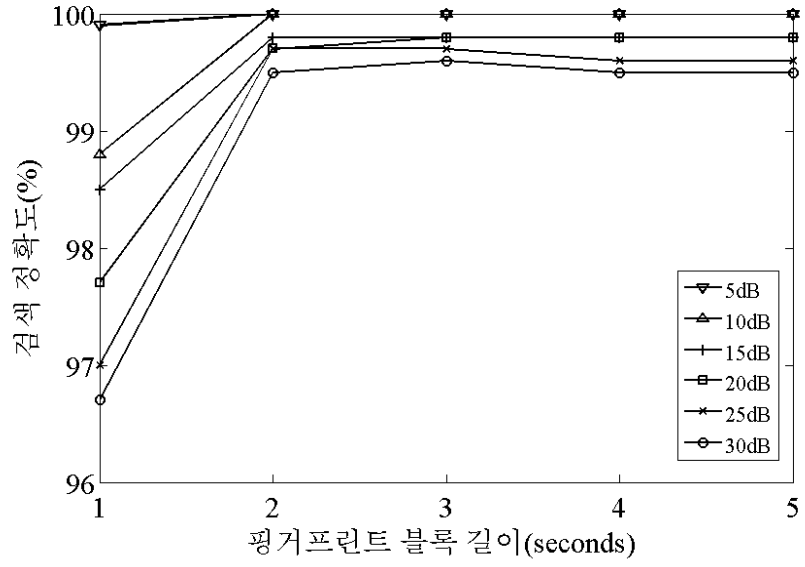
[표 4.9] 검색 샘플별 평균 후보 블록 수 및 후보 리스트 내 위치

	후보 리스트 내 위치	전체 후보 블록 수	위치 비율
SNR 5dB	30	203	15%
SNR 10dB	30	182	17%
SNR 15dB	34	211	16%
SNR 20dB	38	237	16%
SNR 25dB	40	258	16%
SNR 30dB	43	280	15%
평균	36	229	16%

노이즈 샘플에 대해 검색 핑거프린트 길이를 1~5초로 변화하며 “후보 블록 검색” 방식으로 검색한 positive와 negative 샘플의 검색 정확도는 그림 4.10과, 그림 4.11과 같다. Positive 샘플의 경우에는 검색 핑거프린트 블록의 길이가 길어질 수록 정확도가 올라가는 경향이 뚜렷하고, negative 샘플 검색의 경우에는 검색 핑거프린트 블록이 짧고 SNR이 큰 샘플일수록 false positive 결과가 다수 나와 낮은 정확도를 보이지만 3초 이상의 핑거프린트 길이를 사용한 경우에는 높은 정확도를 나타내었다. Negative 샘플의 이 같은 결과는 SNR이 낮을 수록 다양한 악기와 보컬과 같은 음악적인 신호들이 노이즈에 의한 간섭으로 구별하기가 어려워지기 때문에 일반적인 음악과는 전혀 다른 패턴의 핑거프린트가 추출되어 negative 검색 정확도가 높아지게 된다. 반대로 SNR이 높은 경우에는 핑거프린트 추출 시 음악적인 신호의 기여도가 높아지게 되어 원본 음악과 유사한 핑거프린트가 추출되지만 negative검색 시 검색 핑거프린트 블록이 1초인 경우에는 2초 이상을 사용한 경우보다 상대적으로 낮은 정확도를 나타냈다. 이는 검색 오디오 샘플이 1초 정도로 짧은 경우에는 다른 노래임에도 불구하고 비슷한 노래라고 판단할 수 있는 구간이 존재하는 것을 의미하기 때문에 2초 이상의 오디오 샘플을 사용해야 안정적인 검색 정확도를 확보할 수 있다.



<그림 4.10> 검색 핑거프린트 블록 길이에 따른 positive 샘플 검색 정확도



<그림 4.11>검색 핑거프린트 블록 길이에 따른 negative 샘플 검색 정확도

제5장 결론

본 논문에서는 필립스 오디오 핑거프린트 추출 방법을 기반으로 한 최적 특징 파라미터 추출을 위한 실험을 진행하여 오디오 입력 신호가 50~2500Hz, 프레임 길이와 간격이 350ms, 10ms이고 주파수 밴드가 Bark scale 을 가지는 28차 특징에서 가장 좋은 성능을 확인하였다. 또한 기존의 lookup table 기반의 데이터베이스 검색 방법에 대한 성능을 보완하기 위하여 검색 대상에 가중치를 부여하는 방법을 제안하였다. 실험 결과 lookup table 방식의 “전체 검색”과 비슷한 정확도를 나타내었고, positive 샘플의 경우 기존의 lookup table을 이용한 방식에 비해 1.5~4배 정도의 수행 속도 향상이 있음을 확인할 수 있었다. 이는 핑거프린트 차수가 낮은 경우 기존의 검색 방법이 가지고 있는 성능 저하 문제가 상당한 수준에서 개선됨을 의미한다. 또한 음악 검색을 위해서는 최소한 2초 이상의 오디오 샘플을 사용해야 하고 3초 이상의 샘플에서 안정적인 정확도가 나오는 것을 확인하였다. 결과적으로 SNR 10dB 이상의 2~3초 오디오 샘플을 사용할 경우 정확도가 약 80~99%의 음악 검색 성능을 얻을 수 있었다. 그리고 후보 블록 리스트에서 우선순위가 높은 상위 후보군 만을 이용하여 검색을 하게 될 경우 지금보다 성능 향상이 가능할 수 있음을 확인하였다.

참고 문헌

본 논문에서 제안한 개선된 검색 알고리즘은 IEEE Transactions on Consumer Electronics에 제출될 예정이다.

[1] id3.org, http://www.id3.org

[2] IFPI publishes Digital Music Report 2011, http://www.ifpi.org/content/section_resources/dmr2011.html

[3] 이승재, 김성민, 김정현, 유원영, “음악 서비스 및 관련 기술 동향,” 전자통신동향분석 제26권 제2호, 2011년 4월, pp. 148-158

[4] Jin S. Seo, Minho Jin, Sunil Lee, Dalwon, Jang Seungjae Lee, and Chang D. Yoo, “Audio fingerprinting based on Normalized Spectral Sub-band Centroids,” in Proceedings of IEEE ICASSP, Vol.3, 2005, pp. 213-216

[5] Gracenote, http://www.gracenote.com

[6] Shazam, http://http://www.shazam.com

[7] Google, http://www.google.com

[8] Jaap Haitsma, Ton Kalker, “A Highly Robust Audio Fingerprinting System,” in Proceedings of ISMIR 2002, pp. 107-115

[9] Avery Wang, “An Industrial-Strength Audio Search Algorithm,” in Proceedings of ISMIR, Baltimore, MD, Oct. 2003, pp.7-13

[10] Shumeet Baluja, Michele Covell, “Audio Fingerprinting: Combining Computer Vision & Data Stream Processing,” in Proceedings of IEEE ICASSP, Vol2, 2007, pp. 213-216

[11] 이선형, 육동석, “음악 검색을 위한 오디오 핑거프린트 추출 실험,” 한국음향학회 춘계학술발표대회 논문집, 2011, pp.171-172

[12] Barracuda 7200.12 Data Sheet, Seagate, 2011

[13] Memory Technology Evolution: An Overview of System Memory Technologies, Technology brief, 9th Edition, HP, 2010

감사의 글

어느덧 까마득하게만 생각했던 제 논문의 감사의 글을 쓰게 되었습니다. 자신에 대한 부족함으로 시작해서 2년 반 동안 새로운 것에 대한 두려움과 부족했던 시간을 매 순간의 노력과 인내로 이렇게 결실을 맺게 되었습니다. 이러한 결실을 맺기까지 매 순간마다 저를 위해 애써 주신 모든 분들께 이 글을 통해 감사의 마음을 전하고자 합니다.

이렇게 2년 반 동안 끝까지 대학원을 무사히 마칠 수 있도록 용기를 주신 부모님께 감사 드립니다. 그리고 지금까지 경험하지 않은 분야를 논문 주제로 선택하여 학위 논문이 완성되기까지 저에게 많은 지도와 큰 도움을 주신 육동석 교수님께 진심으로 감사의 말씀 드립니다.

대학원 생활을 하는 중에 조언을 해준 고등학교 선배이자 서울대 박사과정에 재학중인 성진이형, 툴 사용에 서툴러 헤맬 때 많은 도움이 된 대학원 동기인 상호형, 힘들어 할 때마다 의지가 되어준 친구 승배, 성익, 재훈, 세원, 인욱, 상준, 민구, 영복, 성길, 준형에게도 감사의 말을 전합니다.

무엇보다도 지난 2년 반 동안 같이 고생한 컴퓨터정보통신대학원 27기 동기들에게 사랑한다는 말을 전하고 싶습니다.

마지막으로 직장 생활과 함께 학업을 병행할 수 있도록 도움을 주신 직장 선후배, 동료들에게도 감사 드리며, 부족하지만 이 논문을 읽는 모든 분들에게 조금이나마 도움이 되기를 바랍니다.