工學碩士 學位論文

다중 특징 벡터를 이용한 고속 오디오 검색

Quick Audio Retrieval Using Multiple Feature Vector

指導敎授 金 基 萬

2005年 2月

韓國海洋大學校 大學院

電 波 工 學 科

潘 知 慧

차 례

기 호 표

C_t	spectral centroid
$D(h_R, h_T)$	Euclidean 거리
$Dc(h_R, h_T)$	Cosine 거리
F_t	spectral flux
h_R	기준 템플릿
h_T	테스트 템플릿
$M_t[n]$	프레임 t 와 주파수 bin n 에서 Fourier Transform의 크기
$N_t[n]$	프레임 t에서 Fourier transform의 정규화된 크기
s	음성 신호의 크기
$S(h_R, h_T)$	h_T 와 h_R 사이의 유사도
$_{ub}S(h_R, h_T(n_2))$	$S(h_R, h_T(n_2))$에 대한 upper bound
Z_i	i 차 zero-crossing rate

ABSTRACT

The types of information are changed text-based into various multimedia data such as speech, image, and moving picture. Therefore, it is necessary to study about searching algorithm. Previous keyword-based retrieval is not optimal for searching the multimedia data. Therefore, the studying is focus on the content-based retrieval (etc. MPEG-7) has been attracted. This thesis concentrated on the content-based retrieval and proposed a quick search method.

In the Audio Information Retrieval (AIR) System, it is important to extract feature vectors. Feature extraction is the process of computing a numerical representation that can be used to characterize a segment of audio. In this thesis, we use the features based on the Short Time Fourier Transform (STFT) and the zero-crossing rates. Firstly, Features based on the STFT are very common and have the advantage of fast calculation based on the Fast Fourier Transform algorithm. The STFT features can be classified into the spectral centroid, the spectral roll-off and the spectral flux. In the second place, the zero-crossing features have been used in the previous papers because of reducing the computation.

This thesis also proposes a new search using the preprocessing and code matching. The previous papers propose a time-series search method using the upper bound proof. It is assumed that similarity between the test and reference template shows considerable correlation from one time step to the next. Because the search algorithm using the upper bound proof computes upper bound on the similarity measures, this method can make possible the quick search. However the search speed of a time-series search method is very low at real time. Therefore this thesis proposes a method using the preprocessing to make up for this defect. Furthermore, we use the code matching method to reduce the matching rates.

This thesis is organized as follows : Section 2 overviews the previous time-series search algorithm. Section 3 explains the core part of our new algorithm and the new optimal combination of multiple features. Section 4 evaluates the accuracy and speed of the algorithm using multiple features. Finally Section 5 gives conclusions and future works.

제 1 장 서 론

컴퓨터와 통신의 발달로 우리가 접하는 정보의 형태는 텍스트(text)에서 점차로 화상, 음성, 동영상 등의 멀티미디어화 및 디지털화하고 있다. 하나의 멀티미디어 데이터는 여러 가지 정보를 포함하고 있으며 다양한 정보를 활용할 수 있는 새로운 기술의 개발과 함께 여러 응용 분야에 이용되고 있다. 이러한 멀티미디어 데이터는 컴퓨터와 통신이라는 매체를 통하여 여러 방법으로 생성되고 저장되며 필요에 따라 탐색 및 검색이 이루어진다. 한편 사용자들에게는 멀티미디어 데이터를 효율적으로 찾아야 하는 필요성이 증가하고 이에 따라 방대한 양의 분산된 멀티미디어 데이터를 처리할 수 있는 색인(indexing) 및 검색 도구의 요구가 커지게 되었다. 멀티미디어 검색 방법에는 크게 두 가지로 분류된다.

첫 번째 방법은 검색의 대상이 되는 모든 멀티미디어 데이터에 사람이 직접 색인을 첨가하고, 사용자 또한 주제어를 이용하여 원하는 정보를 검색하는 텍스트 기반 검색 방법이다. 이 방법은 비정형적인 멀티미디어 데이터에 사람이 수작업으로 의미 정보를 기술하기 때문에 제한된 범위 내에서 효율적인 검색이 가능한 장점이 있으나 대용량의 데이터에 대하여 사람이 일일이 색인을 첨가해야 하기 때문에 시간과 비용이 많이 필요하다. 그리고 멀티미디어가 갖는 복잡한 속성과 특징들을 단순히 텍스트만으로는 정확하게 표현할 수 없는 문제점이 있다.

두 번째 방법은 멀티미디어 데이터의 내용을 대표하는 특징들을

추출하여, 이를 기반으로 검색을 수행하는 내용기반 검색 방법이다. 이 방법은 멀티미디어 데이터로부터 특징들을 자동으로 추출하고 이를 이용하여 검색을 수행하기 때문에 시간 및 인력의 소모를 줄일 수 있다.

기존에는 주로 텍스트 기반의 검색을 통하여 멀티미디어 정보를 검색했으나 최근에 들어서는 텍스트 내용 기반 검색은 한계에 도달한 상황이기 때문에 사용자가 원하는 정보를 보다 정확하게 검색할 수 있는 내용 기반 검색이 주로 사용되고 있다. 하지만 기존의 멀티미디어 표준으로는 이러한 요구사항을 충분히 만족시킬 수 없었고 멀티미디어 데이터를 나타내는 표현을 위한 새로운 노력을 시작하게 되었으며 대표적인 것이 MPEG-7이다. 최근의 기술 발전 추세 및 시장 요구를 바탕으로 하여, 국제 표준화 기구인 ISO와 IEC의 연합기술위원회 산하의 MPEG(공식명칭: ISO/IEC JTC1 SC29/WG11)에서는 MPEG-7 (Multimedia Content Description Interface) 이라는 이름으로 멀티미디어 데이터의 내용기반 검색을 위한 내용 표현 방식에 관한 국제 표준화 작업이 진행되고 있다. 멀티미디어 데이터는 다양한 형태의 데이터들로 구성되어 있으며, 데이터의 크기가 방대하기 때문에 MPEG-7에서는 효과적이고도 효율적인 멀티미디어 데이터 내용 검색을 위한 표현 방법을 표준화하고 있다. MPEG-7에서 데이터 그 자체가 아닌 데이터의 내용에 대한 특징 표현 방법으로 검색하는 표준을 제시하고 있다. 예를 들어 비디오 부분에서 어떤 장면을 찾고 싶을 때 그 장면의 색과 질감(texture) 등의 특성 정보를 이용해 검색을 하고, 오디오도 마찬가지로 노래의 특정 멜로디를 구분하여 검색할

수 있는 것이다. 이 기술은 컨텐츠 자체와 연결되어 사용자가 관심 있는 멀티미디어 자료를 빠르고 효율적으로 찾을 수 있게 한다[1].

비디오 부분에서의 연구로는 IBM사의 QBIC, MIT에서 Photobook, 한국 ETRI에서 BADA_III 와 같은 많은 프로그램들이 있지만, 오디오 부분에서는 Audible Magic에서 나온 Sound Fish라는 프로그램만 있어 상대적으로 오디오 정보 검색에 대한 연구는 미흡한 실정이며 앞으로도 많은 연구가 요구된다.

오디오 데이터에 대한 내용기반 검색 방법에는 크게 브라우징과 인덱스를 통한 검색방법이 있다. 인덱스를 통한 검색 방법은 다시 오디오의 음향이나 음악 등을 분석하여 특징벡터로 인덱스를 만든 후 사용자가 멜로디나 음향 효과로 질의를 하여 원하는 곡을 찾는 방법과 오디오내의 음성을 인식하여 키워드 기반의 인덱스를 만든 후 사용자가 질의를 음성이나 텍스트로 해주는 방법으로 구분할 수 있다.

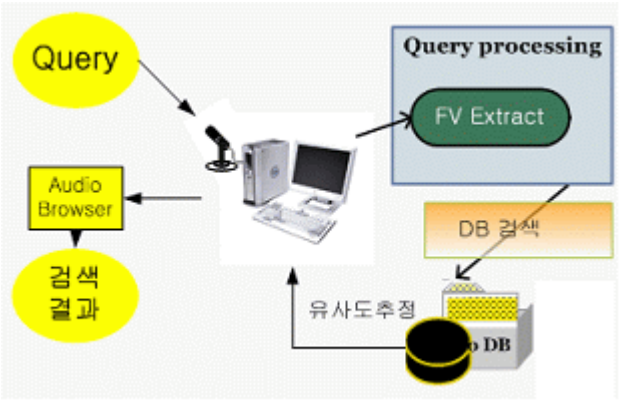

그림 1-1. 내용 기반 검색 과정

Fig. 1-1 The procedure of the content based retrieval.

내용 기반 검색 과정은 먼저 멜로디로 질의를 하여 전체의 오디오로부터 추출한 특징 벡터와 입력한 멜로디의 특징 벡터간의 유사도를 측정하여 원하는 곡을 찾게 된다. 이 과정이 그림1-1에 나타나 있다.

오디오 내용 기반 검색에서의 핵심은 바로 정확한 검색과 고속 검색이 있다. 하지만 두 가지 모두를 충족시키기란 매우 어렵다. 왜냐하면 정확한 검색을 위해서는 많은 특징 벡터가 필요하고 그에 따른 계산량의 증가로 속도가 저하되기 때문이다. 그래서 기존의 논문에서는 특징 벡터를 계산량이 간단한 zero-crossing rate (ZCR) 만을 이용하여 특징벡터를 추출하였지만 본 논문에서는 Short Time Fourier Transform (STFT)에 기반을 둔 centroid, roll-off, flux 와 ZCR을 이용하여 특징 벡터를 구성 할 것이다.

그리고 기존의 논문에서는 upper-bound proof를 이용한 시간 순서 검색을 하였지만, 본 논문에서는 검색에 앞서 전처리를 통하여 검색 속도를 향상시켰다.

본 논문의 제 2장에서는 기존의 논문에서 많이 다뤘던 시간 순서 오디오 검색에 대해 기술하였고, 제 3장에서는 다중 특징 벡터를 이용한 고속 오디오 검색 알고리즘을 제안하였다. 제 4장에서는 제안한 방법의 성능을 비교 하기 위한 시뮬레이션 결과를 나타내었으며, 마지막 5장에서는 결론 및 향후 연구 방향을 제시하였다.

제 2 장 오디오 검색 과정

본 장에서는 기존의 논문에서 주로 사용한 오디오 검색 과정에 대해 설명 하도록 한다. 그림 2-1은 기존의 논문에서 많이 쓰여왔던 오디오 검색 알고리즘이다[2].

검색 방법은 먼저 기준 (reference) 신호와 테스트 신호로부터 특징 벡터들을 추출한 후 각각의 특징벡터들을 히스토그램 모델링을 이용하여 템플릿으로 만든다. 그 다음에 테스트 오디오 stream 에 기준 템플릿을 슬라이딩하면서 서로간의 유사도를 구하는 것이다. 유사도가 어떤 한계점을 넘어버리면 기준 사운드가 감지 되고 그것의 위치를 찾을 수 있게 된다[2]. 이 때 유사도가 한계점을 넘지 않는다면 유사도에 의해 다음의 skip width 가 결정되어진다.

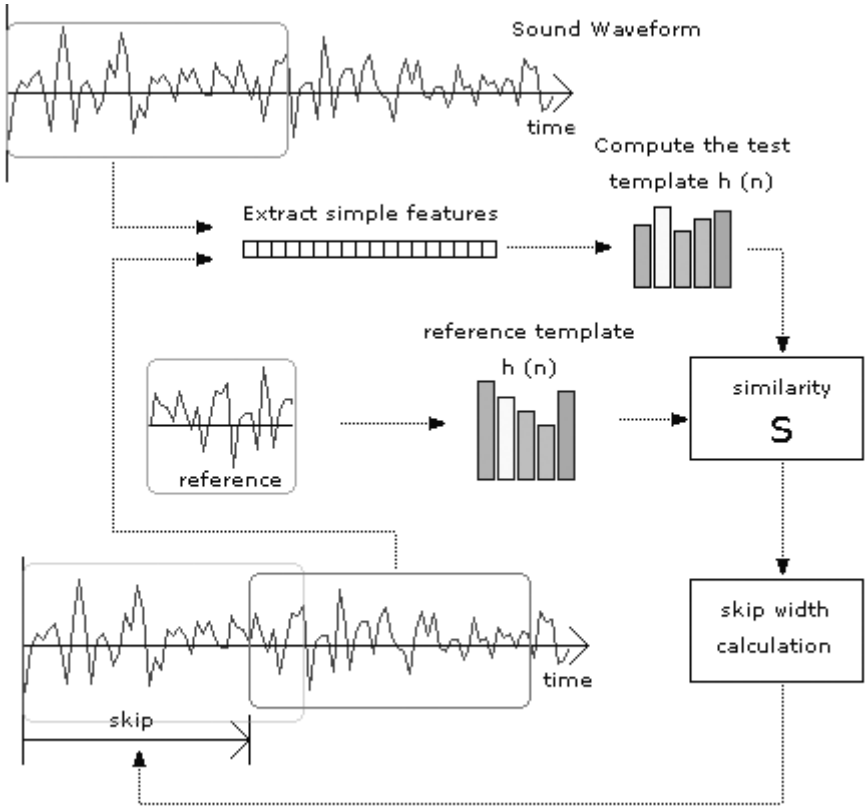

그림 2-1. 오디오 검색 알고리즘

Fig. 2-1 Audio retrieval algorithm.

2.1 특징 벡터 추출 (Feature Vector Extraction)

특징 벡터의 추출은 오디오 신호의 특성을 수치적으로 나타내기 위한 것이다. 오디오로부터 특징 벡터를 추출하는 모든 과정을 “인덱싱”이라고 한다. 인덱싱은 크게 통계적인 특성의 인덱싱과 변환 영역(transform domain) 인덱싱으로 나누어진다. 먼저

통계적인 특성의 인덱싱은 물리적이고 지각적인 특성 들의 평균과 분산 값과 같은 통계적 특성에 의존하는 것을 말하고 변환 영역 인덱싱은 Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Wavelet 과 같은 것을 사용하는 것을 말한다.

특징 벡터는 다양한 오디오 분석과 정보 추출 알고리즘의 근본이 된다. 이 벡터는 전형적으로 하나의 고정된 차원을 가지므로 다차원 특징 공간에서는 하나의 포인트로써 생각 할 수 있다. 오디오에 있어서 특징 벡터를 사용하여 표현할 때는 두 가지 접근이 사용된다.

첫 번째 접근에서 오디오 파일은 시간에 대해 작은 구간으로 나누어져 있고 특징 벡터는 각각의 구간에서 계산 된다. 두 번째 접근에서는 전체 파일에 대한 정보를 요약하는 하나의 특징 벡터를 사용하는 것이다. 첫 번째 접근은 정보가 실시간 업데이트가 필요할 때 적절하다. 예를 들어, mp3 파일들의 자동적인 음악 장르 분류는 두 번째 접근을 사용하여야 하고 라디오 신호의 분류는 첫 번째 접근을 사용해야 한다[4].

오디오 특징 종류로는 기본적인 pitch, loudness, duration, timbre 등이 있고 좀더 상세한 특징으로는 크게 짧은 주기에서 쓰이는 물리적인 특징과 긴 주기에서 쓰이는 지각적인 특징으로 나뉜다. 먼저 물리적인 특징은 주로 스펙트럼의 분포에 따른 것이고 centroid, roll-off, flux 등이 있으며 지각적인 특징에는 rhythm, texture, instrument, MFCC (Mel-Frequency Cepstral Coefficients), LPC (Linear Prediction reflection Coefficients) 등이 있다.

기존의 논문에서는 계산적으로 간단하여 ZCR 만을 특징 벡터로 많이 쓰여왔다. 하지만 본 논문에서는 STFT 에 기반을 둔 특징 벡터를 첨가하여 더 정확한 검색을 할 수 있도록 하였다.

2.1.1 ZCR

ZCR 은 주어진 구간 내에 음성 신호가 기준선인 0 을 통과하는 횟수를 측정하는 것이다. 즉, 이산 신호에서 연속 샘플링 값이 서로 다른 부호일 때 발생하는데, 이는 음성의 분할, 분석, 인식에 매우 유용하게 쓰인다. 특히 적은 계산 양으로 음성이 유성음인지 무성음인지 판단 할 수 있는 중요한 정보를 얻을 수 있어 음성의 실시간 인식에 적용 된다.

음성신호의 크기를 s 라 하고 샘플 n 에 대한 i 차 ZCR Z_i 는 다음과 같이 정의된다.

$$Z_i = \sum_{n=1}^{N} \frac{|\mathrm{sgn}(s_i(n)) - \mathrm{sgn}(s_i(n-1)|}{2} \qquad (2\text{-}1)$$

단, 여기서 $\mathrm{sgn}|s_i(n)|$ 은 다음과 같다.

$$\mathrm{sgn}|s_i(n)| = \begin{cases} 1 & s_i(n) > 0 \\ -1 & s_i(n) < 0 \end{cases} \qquad (2\text{-}2)$$

이와 같은 ZCR 은 원래의 주파수의 대략적인 추정을 하기 위해서 많은 논문에서 사용되어왔다[5].

2.1.2 STFT 에 기반을 둔 특징 벡터

STFT 에 기반을 둔 특징 벡터가 매우 일반적이고 계산을 빨리 할 수 있다는 이점을 지니고 있다. 그래서 본 논문에서는 STFT 에 기반을 둔 특징 벡터들을 이용하였다. 이런 특징 벡터는 centroid, roll-off, flux 로 나눌 수 있다. 각각에 대한 설명은 다음과 같다[4].

Spectral Centroid

spectral centroid 는 STFT 의 magnitude 스펙트럼의 중심을 뜻한다.

$$C_t = \frac{\sum_{n=1}^{N} M_t[n] * n}{\sum_{n=1}^{N} M_t[n]} \qquad (2\text{-}2)$$

여기서 $M_t[n]$ 은 프레임 t 와 주파수 bin n 에서 Fourier Transform 의 크기(magnitude) 이다. centroid 는 스펙트럼의 형태의 측정방법 중의 하나이고 사운드의 날카로움의 정도를

나타내는데 여기서 날카로움이란 스펙트럼의 고주파수 성분과 관련 있다. 그러므로 centroid 값이 더 높을수록 높은 주파수에서 선명한 음질을 나타낸다. 이러한 centroid 는 스펙트럼 형태를 묘사하기 위해 매우 효과적이기 때문에 오디오 분류 작업에서 자주 사용된다[4].

Spectral Roll-off

Spectral roll-off 는 음성 구간과 무 음성 구간 사이를 구분하는 특징으로써 centroid 와 함께 스펙트럼 형태의 또 다른 측정 방법이다. 사실 roll-off 와 centroid 는 매우 관련이 깊다. spectral roll-off 의 정의는 크기(magnitude) 분포의 85%가 집중해있는 주파수 R_t 이하를 말한다.

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 * \sum_{n=1}^{N} M_t[n] \qquad (2\text{-}3)$$

즉, spectral centroid 는 신호의 에너지고 너 낮은 신호에 얼마나 많이 집중되어 있는 가를 보여준다[4].

Spectral Flux

spectral flux 는 스펙트럼의 변화율을 측정하는 방법으로써 연속된 스펙트럼의 분포의 정규화 된 크기들의 차를 제곱한 것을 의미한다.

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2 \qquad (2\text{-}4)$$

여기서 $N_t[n], N_t[n-1]$ 은 각각 현재 프레임 t 와 이전의 프레임 $t-1$에서 Fourier transform 의 정규화된 크기이다. Flux 는 음악과 음성을 구분 짓는데 적합한 특징이다[4].

2.2 히스토그램 모델링과 유사도 측정

기준 사운드를 고정된 길이만큼 나누어서 각각의 프레임으로부터 특징 벡터를 추출하고 특징 벡터들의 분포도를 파악함으로써 기준 템플릿을 얻어진다. 히스토그램은 분포에 대해서 비 모수적 (non-parametric)모델로써 사용된다. 테스트 템플릿도 같은 방법으로 얻어질 수 있다. 테스트템플릿(h_T)과 기준 템플릿(h_R) 사이의 유사도는 히스토그램의 공통부분을 사용하여 다음 식과 같이 얻어질 수 있다[2, 3].

$$S(h_R, h_T) = \sum_{i=1}^{B} \min(h_R^i, h_T^i) \qquad (2\text{-}5)$$

여기서 B는 히스토그램 bin 의 수를 말한다.

2.3 Window Skipping

유사도 측정의 가장 간단한 방법은 기준 템플릿을 슬라이딩해가면서 기준 템플릿과 테스트 오디오 stream 사이의 유사도 측정하는 것이다. 유사도가 일정한 어떤 한계점을 넘어버리면 기준 사운드를 감지하고 그 것의 위치를 찾는 것이다. 이러한 유사도 측정 방법을 “exhaustive 검색”이라고 부르며 이 방법은 매 시간마다 계산을 해야 하므로 계산 량이 많다는 매우 큰 단점을 지니고 있다.

그러나 테스트 템플릿과 기준 템플릿의 현재 시간에서의 유사도는 다음시간에서의 유사도와 영향이 크다는 특성을 지니고 있다. 그렇기 때문에 upper bound 를 이용하여 유사도를 측정하면 이 upper bound 가 감지할 수 있는 한계점을 초과할 때까지의 중간 단계 유사도 측정을 skip 함으로써 계산량을 크게 줄일 수 있게 된다. 히스토그램 모델에 대한 upper bound 는 다음과 같다[2, 3].

Upper bound proof

h_R 은 기준 템플릿 R 에 대한 히스토그램이고 $h_T(n_1)$ 과 $h_T(n_2)$ 는 각각 프레임 n_1 과 n_2 에 대한 테스트 템플릿 T 에 대한 히스토그램이다. 각각 프레임 n_1 과 n_2 에서 히스토그램 유사도를 각각 $S(h_R, h_T(n_1))$, $S(h_R, h_T(n_2))$ 라고 한다. 이 때 $S(h_R, h_T(n_2))$ 에 대한 upper bound 는 다음과 같다.

$$ {}_{ub}S(h_R, h_T(n_2)) = S(h_R, h_T(n_1)) + \frac{(n_2 - n_1)}{N} \quad (2\text{-}6) $$

$$ (n_2 - n_1) = N\{{}_{ub}S(h_R, h_T(n_2)) - S(h_R, h_T(n_1))\} \quad (2\text{-}7) $$

히스토그램 유사도가 어떤 한계점 S_{thresh} 을 초과했을 때, 만약 기준 템플릿과 테스트 템플릿 사이에서 정확한 매치가 이루어진다면, 우리는 그 때의 시간 n_{crit} 에서 히스토그램 유사도를 계산해야 하고 모든 중간의 히스토그램 유사도는 무시하게 된다. n_{crit} 에 대한 식은 다음과 같다[2, 3].

$$ n_{crit} = N\{S_{thresh} - S(h_R, h_T(n_1))\} + n_1 \quad (2\text{-}8) $$

Upper bound proof 의 성능을 검사하기 위해서 30 분짜리 라디오 방송에서 25 분 13 초부터 28 분 35 초에 흘러나온 노래를 검색해보았다. 사용된 특징 벡터는 임의로 ZCR 과 Centroid 로 구성된 벡터를 사용하였다. 실험 결과는 그림 2-2 와 같다.

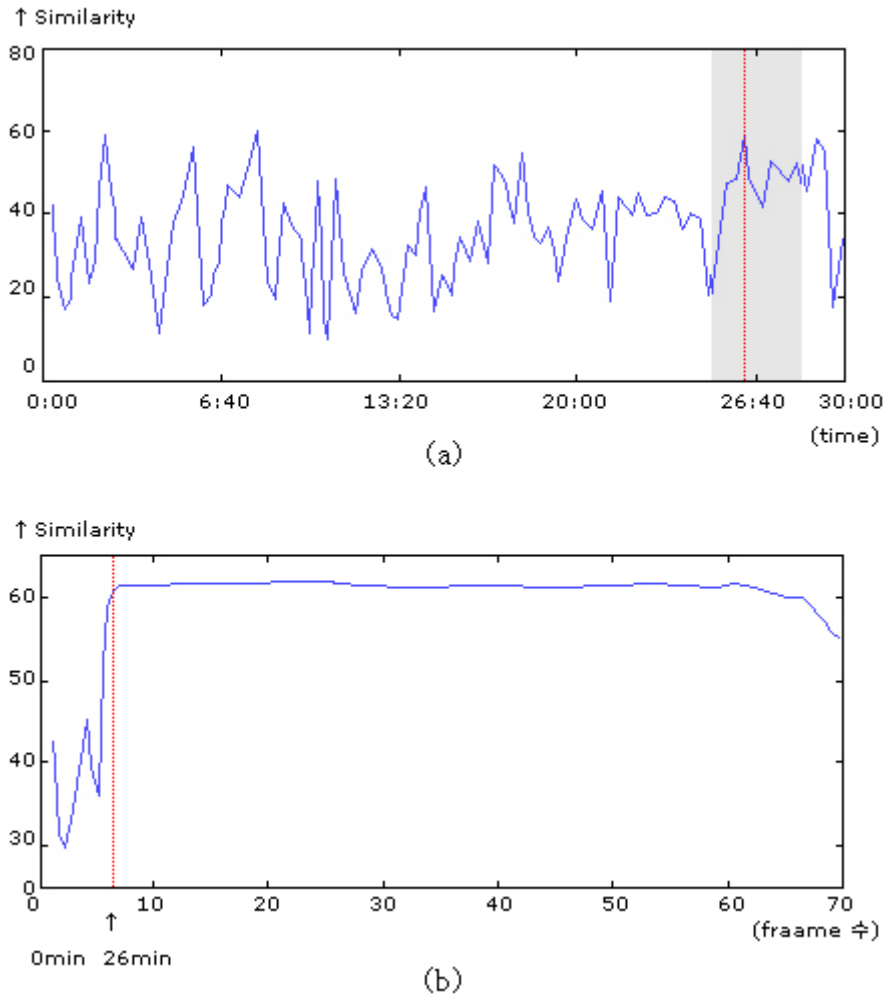

그림 2-2. (a) exhaustive 검색과 (b) upper bound proof를 적용한 검색의 비교

Fig. 2-2 The comparison with (a) exhaustive search and (b) upper bound proof.

먼저 그림 2-2(a)는 앞서 설명한 고정된 프레임을 시간순서대로 검색을 하는 “exhaustive 검색”의 결과이고 그림 2-2(b)는 upper bound proof 를 적용한 결과이다. 먼저 그림 2-2(a)에서는 프레임이 80 번째가 되어서야 사운드의 위치를 감지할 수 있었고, 총 검색 시간은 17 분이라는 시간이 걸렸다. 하지만 upper bound proof 를 적용시켰을 때는 단지 6 번째 프레임에서 사운드의 위치가 감지되었고 검색시간은 1 분 35 초가 걸렸다. upper bound proof 를 사용하였을 때 검색 속도가 향상되었다는 것을 알 수 있다[6].

2.4 시간 순서 오디오 검색의 단점

기존의 논문에서 많이 다뤄왔던 시간 순서 오디오 검색은 많은 단점을 가지고 있다. Upper bound proof 로 검색시간을 향상시켰지만 30 분짜리 오디오 stream 에서 찾고자 하는 사운드의 위치를 검색하는 시간이 1 분 35 초는 결코 짧은 시간이 아니다. 현재 많이 쓰이는 텍스트 위주의 기반 검색에서 걸리는 시간은 몇 초도 걸리지 않는다. 여기에 익숙해있던 많은 사람들에게는 1 분 35 초라는 시간은 너무나 길고 지루하게 여겨질 것이다. 그러므로 그 단점을 보완해줄 수 있는 것이 바로 본 논문에서 제안하는 방법이다. 그 방법은 다음 3 장과 4 장에 걸쳐 설명을 하도록 한다.

제 3 장 다중 특징 벡터를 이용한 고속 오디오 검색

기존의 오디오 검색 방법은 하나의 특징 벡터만을 이용하였기 때문에 정확도가 떨어지고, 시간 순서로 검색을 하기 때문에 계산량이 늘어서 검색 속도가 떨어지는 것을 2 장으로부터 알 수 있었다. 본 장에서는 이러한 단점을 보완하기 위해 새로운 고속 오디오 검색 방법을 제안하였다. 본 논문에서 제안하는 고속 오디오 검색은 그림 3-1 과 같다.

이 방법은 2 장에서의 기존 논문에서의 검색 방법과는 달리 전처리 단계를 두어서 검색을 효율적으로 할 수 있게 하였다. 전처리 단계를 살펴보면, 먼저 오디오 stream 을 고정된 길이의 프레임으로부터 특징 벡터를 추출한 후, 히스토그램 모델링을 한다. 이 때 각각의 히스토그램의 패턴에 따라서 하나의 히스토그램은 하나의 코드를 가지게 된다. 이러한 코드는 주파수에 따른 특징 벡터들의 분포를 미리 짐작해볼 수 있고 많은 데이터들로부터 검색을 효율적으로 할 수 있게 도와준다.

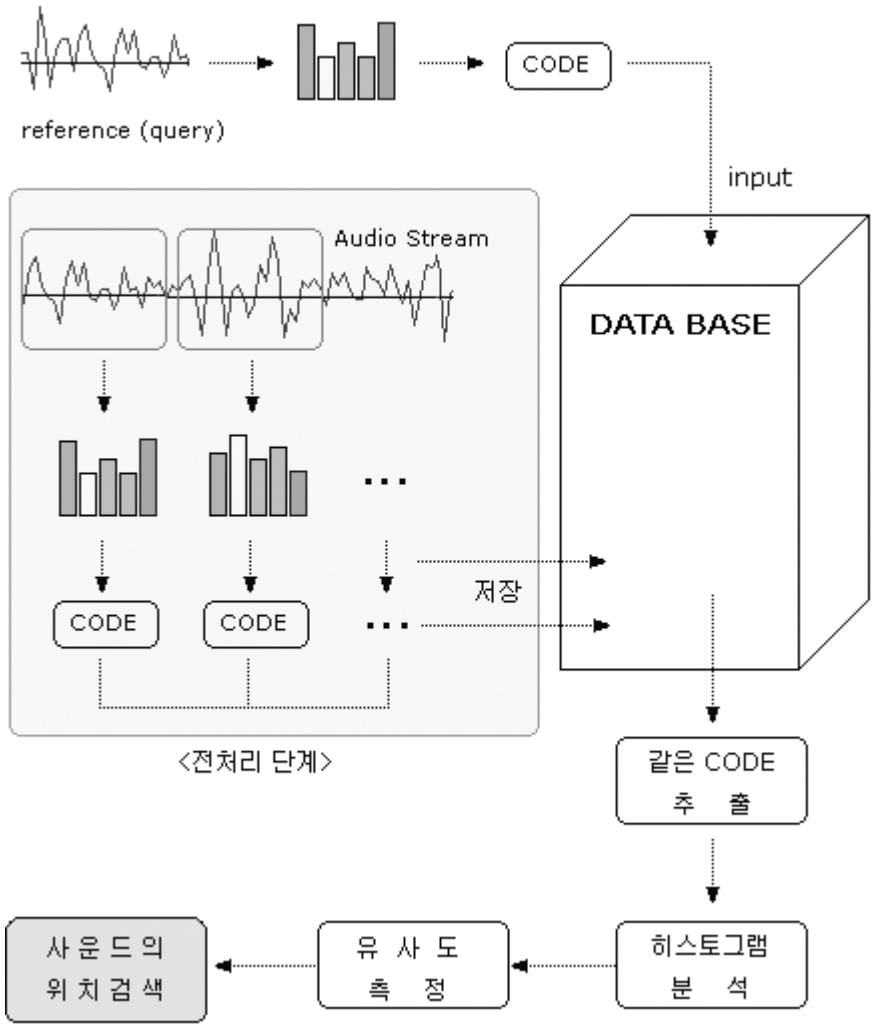

그림 3-1. 고속 오디오 검색 시스템

Fig. 3-1 Quick audio search system.

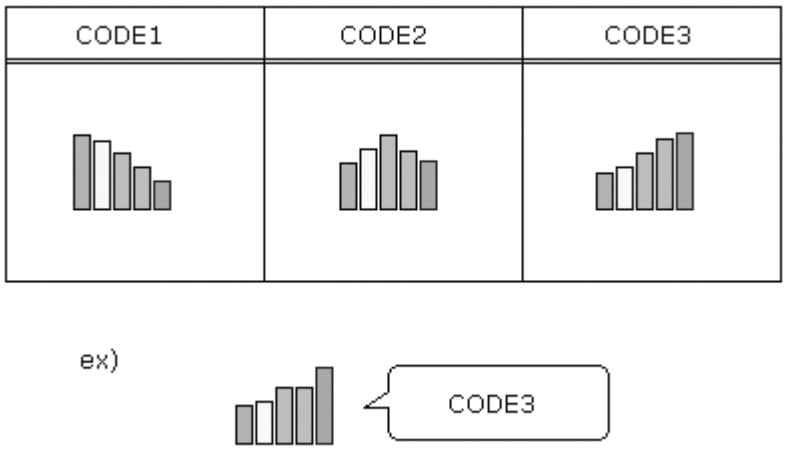

그림 3-2. 히스토그램 코드의 예

Fig. 3-2 The example of the histogram code.

그림 3-2 는 코드에 대한 간단한 예를 나타낸 것이다. 이 그림에서는 히스토그램의 모양에 따라서 CODE 1 과 CODE 2, 그리고 CODE 3 로 분류했다. 어떤 하나의 히스토그램이 구해지면 미리 정의해둔 코드와 가장 유사한 형태의 코드 번호를 가지게 되는 것이다. 그림 3-2 에서 예로 든 히스토그램은 CODE 3 의 히스토그램과 가장 유사하기 때문에 코드명을 CODE 3 으로 가지게 되는 것이다.

코드가 정해지면 각각의 프레임의 히스토그램과 코드를 데이터베이스에 저장하게 된다. 지금까지의 단계가 바로 전처리 단계이다.

두 번째 단계는 질의를 입력하는 단계이다. 만약 어떤 음악을 검색하기 위해서 그 음악의 일부분을 입력으로 넣어주면 기준

파일로부터 히스토그램과 그것의 코드를 구한다. 이 코드를 가지고 데이터 베이스로부터 같은 코드를 추출하게 된다. 선정된 코드의 프레임을 가지고 히스토그램을 다시 분석하여 유사도를 측정하여 어떤 한계치를 넘어버리면 그 사운드의 위치가 감지되는 것이다.

이 알고리즘은 전체의 오디오 stream 을 각각 유사도를 측정할 필요가 없기 때문에 계산적으로도 간단하고 미리 전처리 단계를 두기 때문에 검색 시간이 매우 짧다는 이점을 가진다.

3.1 다중 특징 벡터 구성

기존의 방법에서는 주로 계산적으로 간단한 ZCR 만으로 이루어진 단일 특징 벡터를 이용하여 오디오 검색을 하였다. 본 논문에서는 보다 정확하고 빠른 검색을 위해서 ZCR 과 2 장에서 설명한 STFT 에 기반을 둔 centroid, roll-off, flux 를 이용하여 다중 특징 벡터를 구성하려고 한다.

표 3-1. 특징 벡터들의 조합

Table 3-1. The combination of feature vectors.

1 개로 구성	2 개로 구성	3 개로 구성	4 개로 구성
centroid	centroid + roll-off (CR)	centroid + roll-off + ZCR (CRF)	centroid + roll-off + flux + ZCR
roll-off	centroid + flux (CF)	centroid + roll-off + ZCR (CRZ)	
flux	centroid + ZCR (CZ)	centroid + flux + ZCR (CFZ)	
ZCR	roll-off+ flux (RF)	roll-off + flux + ZCR (RFZ)	
	roll-off+ ZCR (RZ)		
	flux + ZCR (FZ)		

이러한 벡터들이 어떻게 조합이 될 때 가장 좋은 성능을 나타내는지, 가장 빠른 계산을 하는지 비교하기 위해서 이 4 개의 특징 벡터들로 만들 수 있는 15 개의 조합을 만들었다. 그 조합은 표 2-1 과 같다.

3.1.1 다중 특징 벡터 조합의 정확도 비교

각각의 특징 벡터들의 성능을 비교 하기 위해서 두 가지의 테스트를 수행하였다. 먼저 첫 번째 테스트는 정답인 지점에서 매칭율을 통해서 원하는 구간에서 얼마나 정확하게 검색이 되는

지를 알아보았고, 두 번째 테스트는 어떠한 한계치를 넘을 때 후보자의 매칭율을 통하여 어떤 특징 벡터의 구성에서 후보자의 수가 감소되는 지를 알아보았다. 테스트에 앞서 장르를 크게 발라드, 댄스, 락, 이렇게 세가지 종류로 나누었고 이 세가지 종류에 따라 비교 분석하여 장르와 상관없이 가장 최적화된 특징 벡터의 구성을 알아보려는 것이 이 테스트의 목표이다.

먼저 장르별로 다양한 노래가 나오는 라디오 프로그램을 선정한 뒤, 그 방송을 11.025kHz 로 1 시간 녹음하여, 그 오디오 stream 에 들어있는 특정한 노래를 검색하도록 하였다. 이 때 질의를 할 곡은 장르별로 각각 세 곡을 간추려 테스트를 해보았다. 먼저 질의를 위해 검색할 곡들의 O.S.T 를 구하여 그 노래마다 20 초간 추출하여 웨이브 파일로 만들었다. 즉, 테스트템플릿이 바로 길이가 1 시간인 오디오 stream 이고 기준 템플릿은 질의를 위해 20 초간 추출한 웨이브 파일이 된다. 이 기준 템플릿을 테스트 템플릿에 슬라이딩 해가면서 각각의 특징 벡터 값을 구하게 된다. 이 때의 테스트 템플릿과 기준 템플릿의 각각의 특징 벡터 값의 거리를 구하게 되고 이 거리 차가 가장 작게 나타나는 지점이 바로 결과 지점인 것이다.

Test 1

첫 번째 테스트에서는 정답인 지점에서의 매칭율을 나타낸다. 다시 말하면 정답인 지점을 검출 할 수 있는 비율을 나타낸 것이다. 그 결과는 그림 3-3 에서 보여준다. 가로축은 각각의 특징 벡터의

조합을 말하고 세로축은 매칭율을 말한다. (a), (b), (c) 각각은 발라드, 댄스, 락 장르이고 마지막 (d)는 총 평균을 나타낸 것이다.

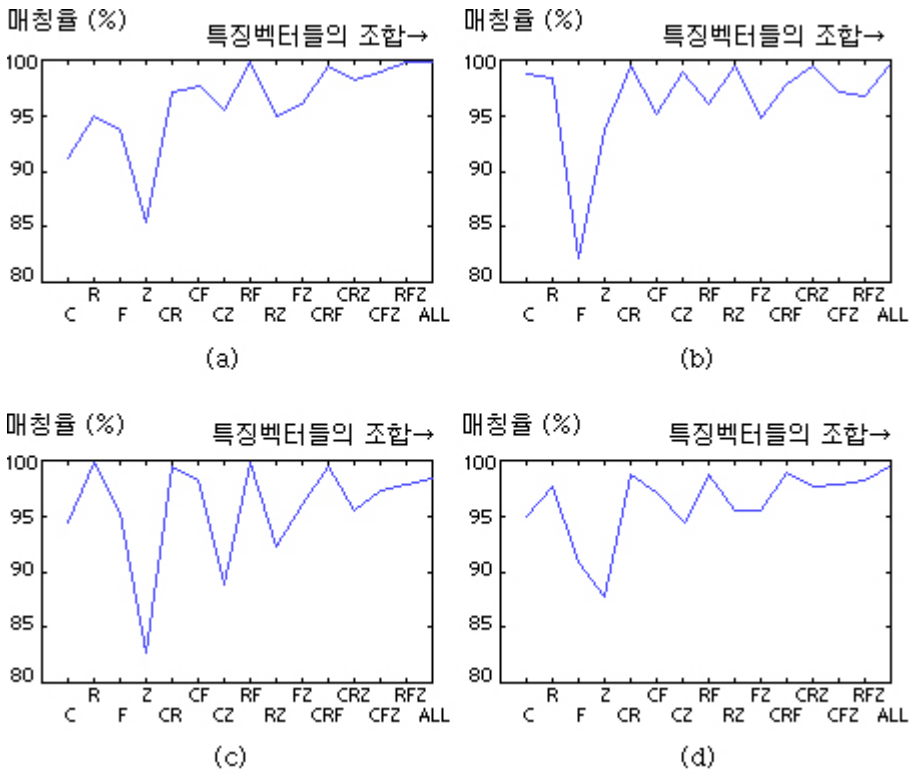

그림 3-3. 장르별 매칭율 비교 분석

(a) 발라드 (b) 댄스 (c)락 (d) 총평균

Fig. 3-3 The comparative analysis of matching rate for each genre

(a) ballad (b) dance (c) rock (d) total average.

이 매칭율이 클수록 검색의 정확도가 크다고 할 수 있다. 가장 이상적인 검색은 바로 이 매칭율이 100%가 되는 것이지만

실제적으로는 노래마다 비슷한 구간이 있을 수도 있기 때문에 100%의 검색 정확도를 만들기는 매우 힘들다. 하지만 이와 같은 다중 특징 벡터의 조합으로 정확도를 높일 수 있다.

기존의 논문에서 주로 쓰였던 ZCR 만을 특징 벡터로 두고 검색을 했을 때 매칭율은 다른 특징 벡터들의 구성과 비교해 볼 때 월등한 차이가 보였다. 하지만 ZCR 도 다른 특징 벡터들과의 결합하면서 매칭율이 매우 향상 된 것을 알 수 있다. 이와 같은 경우는 다른 특징 벡터들에서도 나타난다. centroid 나 roll-off, 그리고 flux 같은 경우도 다른 특징 벡터들과 결합하면서 매우 좋은 향상을 보였다.

각각의 장르별로 다른 점도 있지만 공통적으로 'roll-off + flux', 'centroid + roll-off + flux', 'centroid + flux + ZCR'. 'roll-off + flux + ZCR', 그리고 모든 특징 벡터로 다 구성하였을 때 좋은 결과가 나왔다는 것을 그림을 통해 알 수 있다.

Test 2

두 번째 테스트에서는 거리가 0.03 이하인 지점의 매칭율을 구해보았다. 정확한 검색을 위해서는 이 구간의 매칭율이 낮아야 한다. 만약 매칭율이 높다면 그만큼 후보자 수가 많으므로 정확한 검색에 방해를 주게 된다. 이 실험의 결과는 그림 3-4와 같다. 이 그림에서는 하나의 특징 벡터를 이용하는 것 보다 여러 가지 특징 벡터를 이용할 때 매칭율이 낮아지는 것을 알 수 있다. 이 그림에서도 장르별로 조금은 다르지만 공통적으로 'flux + ZCR',

‘centroid + roll-off + flux’, ‘centroid + flux + ZCR’, ‘roll-off + flux + ZCR’, 그리고 모든 특징 벡터를 사용할 때 좋은 결과가 나왔다는 것을 알 수 있다.

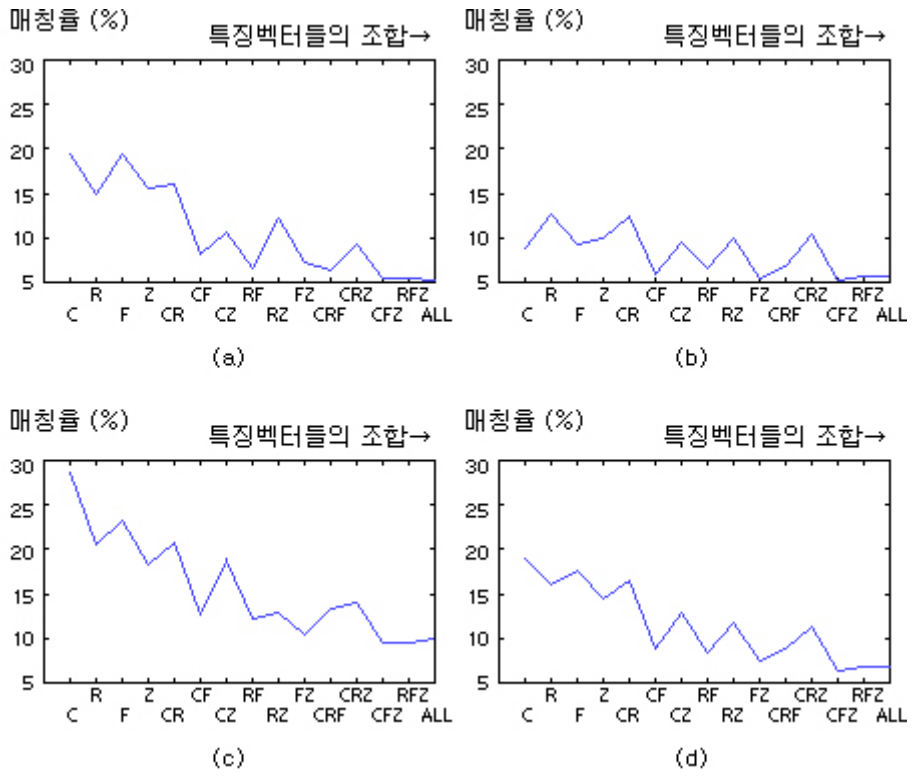

그림 3-4. 장르별 거리가 0.03 이하인 지점의 매칭율 비교 분석

(a) 발라드 (b) 댄스 (c) 락 (d) 총평균

Fig. 3-4 The comparitive anlysys of matching rate below 0.03(distance) for each genre.

(a) ballad (b) dance (c) rock (d) total average

표 3-2. 특징 벡터들의 성능 평가 (W:나쁨, G:좋음, B:매우 좋음)

Table 3-2. The performance evaluation of feature vectors.

(W: Worst, G: Good, B:Best)

특징 벡터들의 조합	발라드	댄스	락
centroid	W	B	W
roll-off	W	G	W
flux	W	W	W
ZCR	W	W	W
centroid+ roll-off	G	G	G
centroid+ flux	B	G	G
centroid+ ZCR	G	G	W
roll-off+ flux	B	G	B
roll-off+ ZCR	W	G	W
flux+ ZCR	B	G	G
centroid+ roll-off+ flux	B	B	B
centroid+ roll-off+ ZCR	G	G	W
centroid+ flux+ ZCR	G	B	G
roll-off+ flux+ ZCR	B	B	B
All	B	B	B

두 가지의 테스트들을 통하여 특징 벡터들의 성능을 평가하였다. 그 결과는 표 3-2에 나타나 있다. W는 나쁨을 의미하고 G는 좋음, 그리고 B는 매우 좋음을 뜻한다. 여기서 W, G, B의 기준은 첫 번째 테스트에서 매칭율이 100%일 때 5점, 96~99%일 때 4점, 81~95%일 때 3점, 76~80%일 때 2점, 71~75%일 때 1점, 그 외에는 0점으로 하였고 두 번째 테스트에서는 매칭율이 0~5%일 때 5점, 6~10%일 때 4점, 11~15%일 때 3점, 16~20%일 때 2점, 21~25%일 때 1점, 그 이상의 매칭율이 보일 때는 0점으로 하여 점수화시켜 첫 번째 테스트 결과와 두 번째 테스트 결과를 합쳐서 100점 만점으로 점수를 나타내어 50점 밑으로는 W, 60~70점은 G, 80~100점은 B로 하였다. 앞의 두 가지의 테스트를 통해서 여기서 각 장르별로 약간의 차이는 있지만 공통적으로 'centroid + roll-off + flux'와 'roll-off + flux + ZCR' 그리고 모든 벡터를 다 사용하였을 때 가장 좋은 성능이 나온 것을 알 수 있다.

3.1.2 다중 특징 벡터 조합의 처리 시간 비교

그림 3-5 는 각각의 특징 벡터들을 구하는데 걸리는 시간을 나타낸 것이다. 이 테스트는 같은 환경에서 수행하였고 centroid가 걸리는 시간을 1로 잡아서 각각의 특징 벡터들의 처리 시간을 구한 것이다. 이 실험에서 기존에 계산량이 적어서 논문에 많이 응용된 ZCR이 다른 특징 벡터들에 비해 처리하는데 걸리는 시간이 많이 걸리고 STFT에 기반을 둔 다른 특징 벡터들은 처리 시간이 거의

비슷했다. 그러므로 ZCR이 포함된 벡터들의 처리시간은 상당히 많이 든 반면, 그것이 포함되지 않은 'centroid + roll-off + flux'는 세 가지의 조합이지만 처리 시간이 매우 적게 드는 것을 알 수 있다.

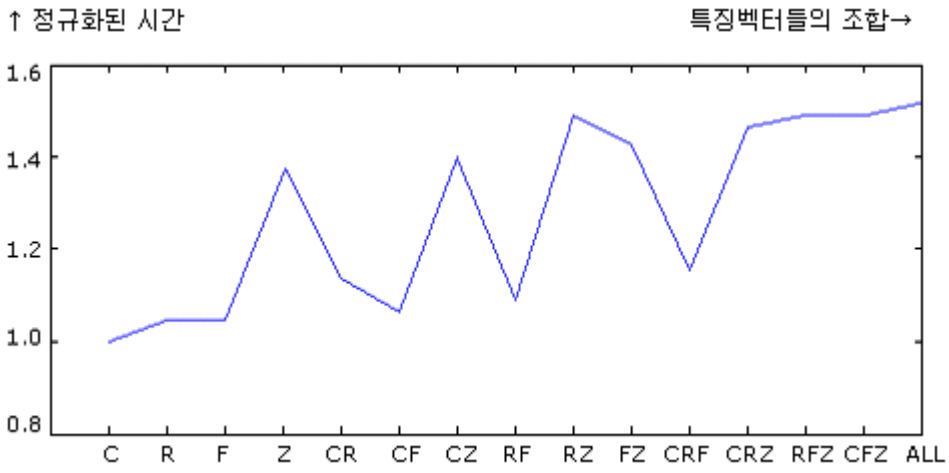

그림 3-5. 각각의 특징 벡터를 구하는데 걸리는 시간

Fig. 3-5 Processing time for calculating the feature vectors.

3.1.3 다중 특징 벡터의 조합

기존의 논문에서는 계산량의 문제로 ZCR만을 특징 벡터로 사용하였지만 본 논문에서는 보다 정확한 검사를 위하여 다중 특징 벡터를 사용하였다. 본 논문의 목적은 바로 보다 정확하고 빠른

검색에 있다. 그러므로 정확하고 빠른 검색을 위하여 정확도와 처리시간 테스트를 수행하였다. 그 결과 정확도 테스트에서 'centroid + roll-off + flux'와 'roll-off + flux + ZCR' 그리고 모든 벡터를 다 사용한 것의 성능이 가장 좋았고 처리시간 테스트에서는 ZCR을 제외한 특징 벡터들의 구성에서 좋은 결과가 나왔다. 본 논문에서는 정확하고 빠른 검색, 이 두 가지를 충족시키기 위해 'centroid + roll-off + flux'로 구성된 특징 벡터를 사용하였다.

그림 3-6은 기존의 논문에서 주로 사용한 ZCR과 본 논문에서 사용할 'centroid + roll-off + flux'로 구성된 특징 벡터의 성능을 비교한 그림이다. 30분 짜리 오디오 stream에서 20초 짜리의 곡을 기준 사운드로 두어 질의를 하도록 하였다. 기준 사운드의 특징 벡터와 오디오 stream의 특징 벡터간의 거리를 구해보았다. 질의 곡의 위치는 24분 55초에서 28분 57초 사이이며 그 구간은 음영을 준 구간이며, 기준 사운드가 위치해있는 구간은 25분 40초에서부터 26분에 위치해 있으며 빨간 점선으로 표시했다. ZCR로만 구성된 특징 벡터를 이용했을 때는 정답 구간이 아닌 곳에서 더 낮은 값이 나왔지만, 'centroid + roll-off + flux'로 구성된 특징 벡터를 이용했을 때는 정확하게 원하는 구간에서 가장 낮은 값이 나왔다. 그리고 'centroid + roll-off + flux'로 구성된 특징 벡터를 이용한 검색 시간은 ZCR로만 구성된 특징 벡터를 이용할 때보다 더 단축할 수 있었다. 물론 다른 곡을 기준 사운드로 두었을 때의 실험에서도 정확도와 검색 시간에서 매우 향상된 것을 알 수 있었다. 그것으로 보아 'centroid + roll-off + flux'로 구성된 특징 벡터는 성능면에서도 시간면에서도 ZCR보다 월등하기

때문에 정확하고 빠른 고속 검색에서 적합한 특징벡터의 조합이라고 할 수 있다.

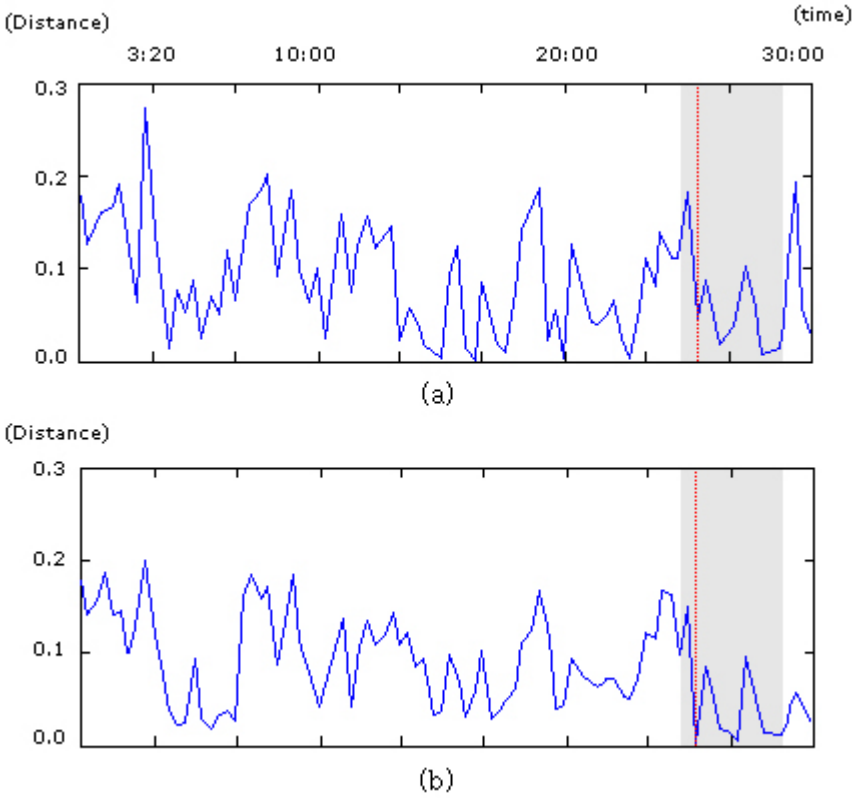

그림 3-6. (a) ZCR과 (b) centroid + roll-off + flux의 거리 비교

Fig. 3-6 The comparison with each distance of (a) ZCR and (b) centroid + roll-off + flux.

3.2 유사도 측정

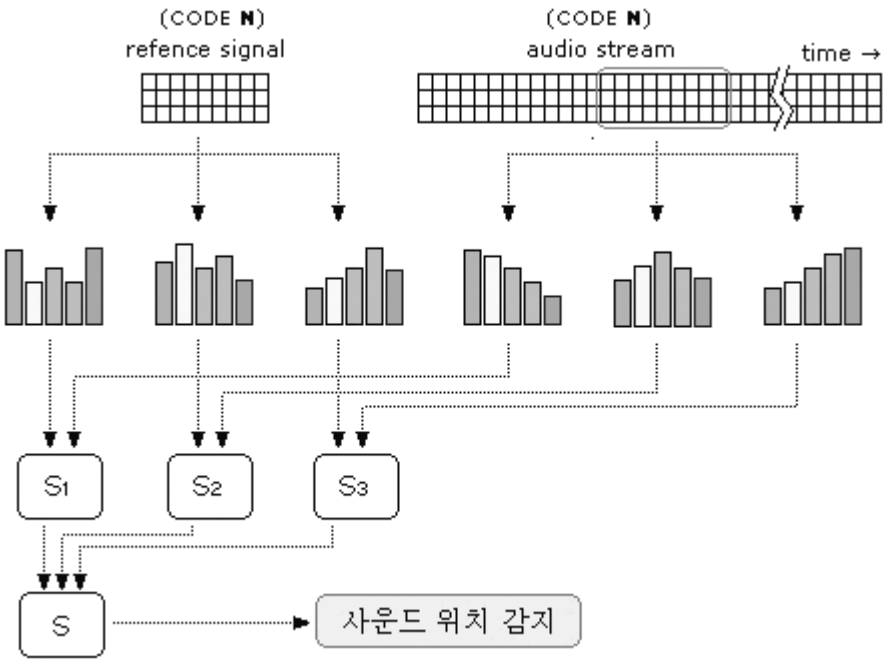

그림 3-7. 유사도 측정

Fig. 3-7 Similarity measurements.

기준 신호의 위치를 감지하기 위해서는 먼저 기준 신호의 특징 벡터를 추출한다. 본 논문에서는 Centroid, Roll off, Flux로 이루어진 특징 벡터를 구성할 것이다. 각각의 특징벡터는 그림3-7처럼 히스토그램 모델링을 하여 코드를 구하게 된다. 그 다음에는 데이터 베이스로부터 같은 코드를 가진 프레임을 구하게 되는 것이다. 그림 3-7에서도 기준 신호가 CODE N 이라고 한다면

데이터 베이스로부터 CODE N 을 가진 프레임을 가지고 와서 그 프레임의 히스토그램과 기준 신호의 히스토그램의 유사도를 측정하게 되는 것이다.

3.2.1 여러가지 유사도 측정 방법

유사도를 측정하기 위해서는 많은 방법이 제시되었다. 먼저 2장에서 제시한 일반적인 유사도 측정 방법과 Euclidean Distance 측정 방법, 그리고 Cosine Distance 측정 방법이 있다.

Euclidean Distance

$$D(h_R, h_T) = \sqrt{\sum_{i=1}^{B} (h_R^i - h_T^i)^2} \tag{3-1}$$

의 값은 낮을수록 유사도가 높다고 할 수 있다[7].

Cosine Distance

$$Dc(h_R, h_T) = \frac{\sum_{i=1}^{B} h_R^i h_T^i}{\sqrt{\sum_{i=1}^{B} (h_R^i)^2 \times \sum_{i=1}^{B} (h_T^i)^2}} \tag{3-2}$$

Cosine Distance 방법은 기존의 텍스트 문서의 유사도를

측정하는데 주로 사용되어져 왔다. Cosine Distance는 Euclidean Distance와는 달리 값이 높을수록 유사도가 높다고 할 수 있다[7].

3.2.1 유사도 측정 방법에 따른 성능 비교

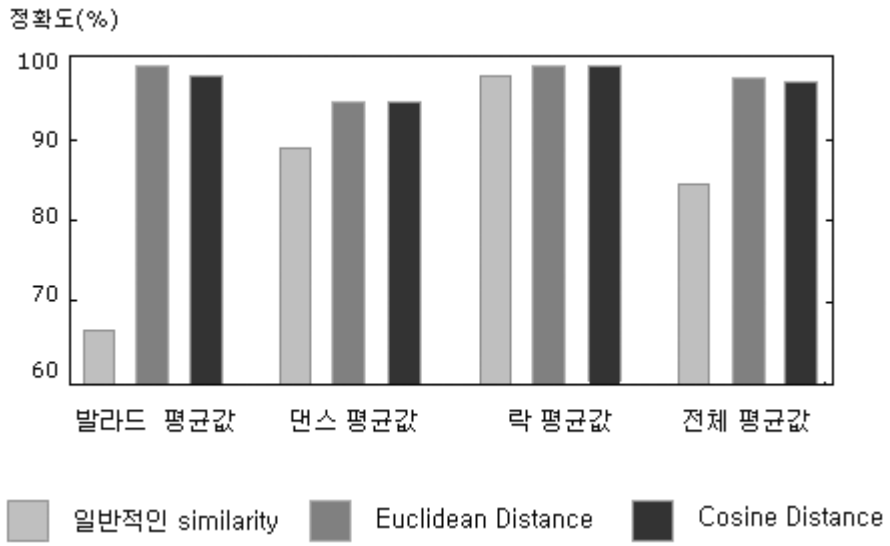

그림 3-8. 각각의 유사도 측정에 따른 정확도 비교

Fig. 3-8 The comparison of the accuracy for similarity measurements.

다중 특징 벡터를 이용하였을 때 가장 적합한 유사도 측정 방법을 비교하기 위해서 위에서 제시한 세 가지의 유사도를 측정하였다. 30분 짜리 오디오 stream에서 20초 짜리의 곡을 기준 사운드로 두어 질의를 하도록 하였다. 이 실험은 장르별로 5곡을 선정하여

유사도를 측정하고 원하는 구간에 얼마나 정확하게 검색을 하는지 알아보았다. 이 실험 결과는 그림 3-8에 나와 있다. 장르에 상관없이 Euclidean Distance와 Cosine Distance의 결과가 잘 나오는 반면, 일반적인 유사도의 결과는 발라드에서는 매우 부진하게 나왔고 전체 평균도 다른 측정 방법에 비해서 만족할만한 결과가 나오지 않았다. 정확도의 전체 평균값은 일반적인 유사도를 이용하였을 때는 84.4%, Euclidean Distance를 사용하였을 때는 97.4%, Cosine Distance를 사용하였을 때는 97.0%라는 결과가 나왔다.

그림 3-9는 이 실험에서 임의의 한 곡에 대한 전체 유사도의 결과 값이다. 이 노래 전체의 위치는 24분 55초에서 28분 57초 사이이며 그 구간은 음영을 준 구간이며, 질의를 했던 기준 신호가 위치해있는 구간은 25분 40초에서부터 26분에 위치해 있으며 빨간 점선으로 표시했다.

이 실험 결과, 일반적인 유사도 측정에서는 점선 외의 다른 구간에서도 높은 값이 나와서 검색의 정확도가 매우 떨어져 있는 것을 알 수 있다. 그리고 Euclidean Distance와 Cosine Distance는 원하는 구간에서는 높은 값이 나왔지만 Cosine Distance에서는 후보자들의 수가 너무 많다는 것을 알 수 있다. 그러므로 본 논문에서는 유사도 측정을 검색의 정확도가 높고 후보자 수가 적은 Euclidean Distance로 정확한 검색을 하도록 하였다.

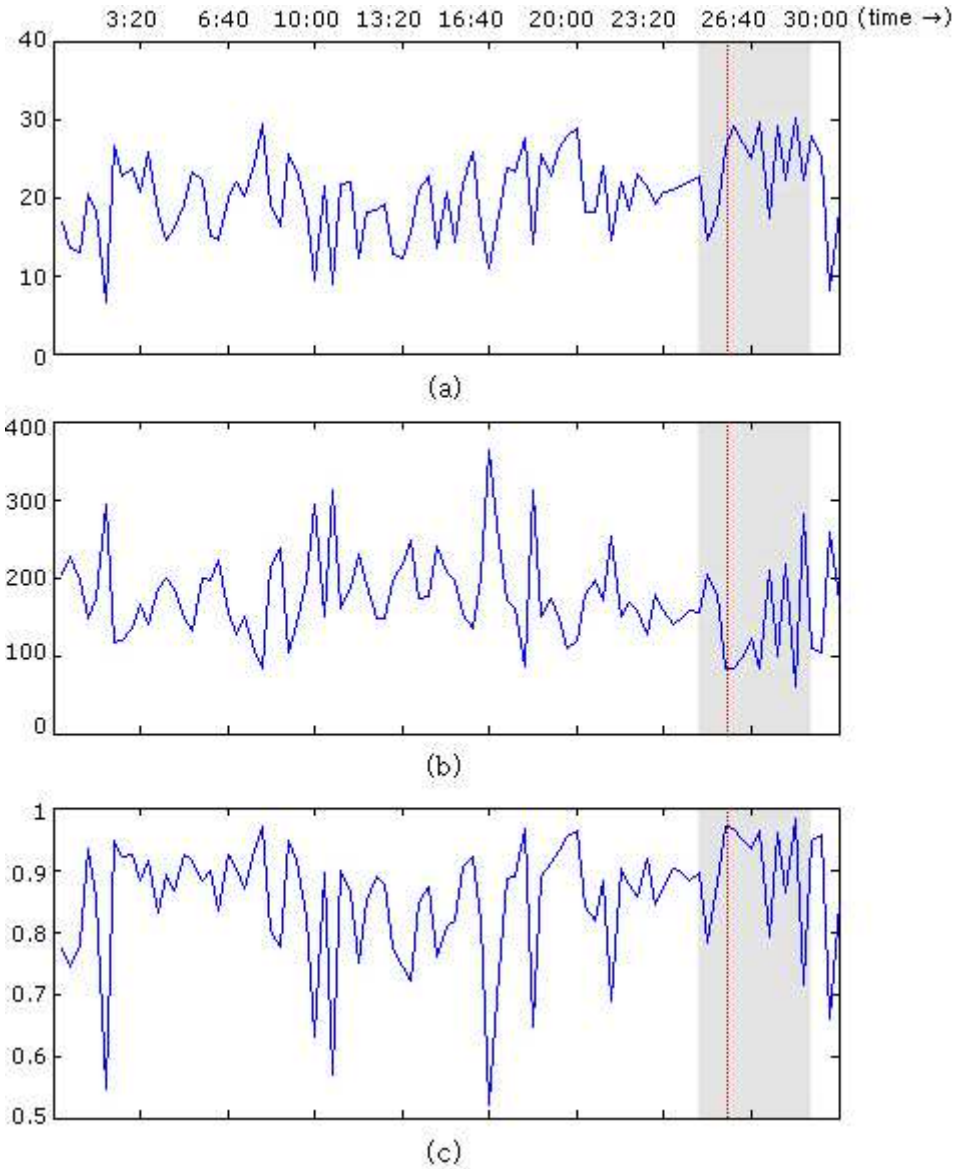

그림 3-9. 유사도 측정 비교 분석

(a) 일반적인 유사도 측정 (b) Euclidean Distance (c) Cosine Distance

Fig. 3-9 The comparative analysis of similarity measurements

(a) General similarity (b) Euclidean Distance (c) Cosine Distance.

본 논문에서 제안한 고속 오디오 검색의 알고리즘을 정리하면 다음과 같다.

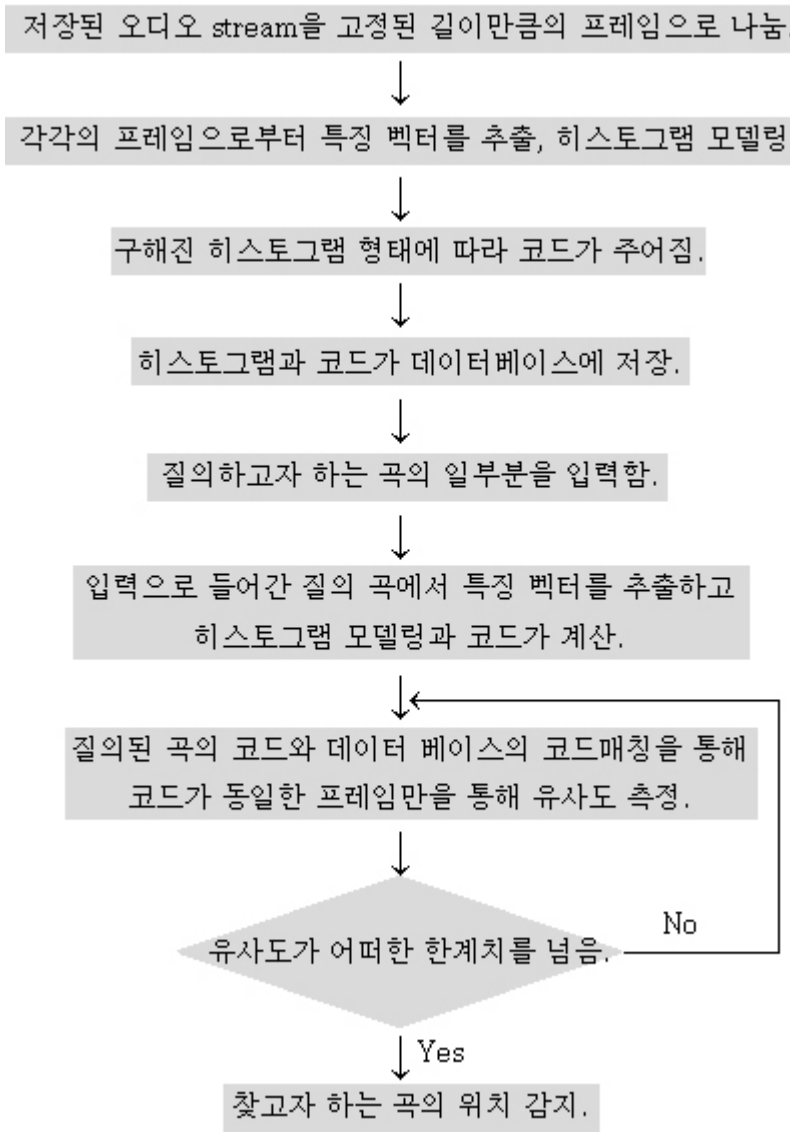

그림 3-10. 제안된 방법의 흐름도

Fig. 3-10 Flow chart of the proposed method.

제 4 장 실험 과정 및 결과

본 논문에서 제안한 검색의 방법의 성능을 알아보기 위해 3 시간 짜리 라디오 방송을 11.025 kHz 로 녹음하였고, 방송에서 2 시간 8 분 49 초부터 2 시간 12 분 18 초에 흘러나온 JNC 의 “하루가 지나고”라는 곡의 위치를 검색하였다. 검색에 앞서 찾고자 하는 노래의 O.S.T 를 찾아 그 노래의 20 초 만 추출하여 기준으로 두어 실험을 하였다.

4.1 검색의 정확도

기존의 논문에서는 주로 ZCR 만을 사용하였지만, 본 논문에서는 검색의 정확도를 높이기 위해서 다중 특징 벡터를 사용하여 검색을 하였다. 여기에서 쓰인 특징 벡터는 3 장의 여러 가지 테스트를 통해서 정확도와 속도에서 가장 우수한 성능이 나왔던 Spectral centroid, roll-off, flux 를 이용하여 다중 특징 벡터를 구성하였다. 구성한 다중 특징 벡터를 이용하여 히스토그램 모델링을 하여 각각의 히스토그램의 Euclidean Distance 를 구하였다. 물론 가장 거리가 작게 나온 값이 바로 우리가 원하는 지점이다. 그 결과는 그림 4-1 에 나타나 있다. 점선으로 표시된 부분이 바로 기준 사운드가 위치해 있는 지점이다. 실험 결과, 원하는 지점에서 정확하게 위치를 감지할 수 있었다.

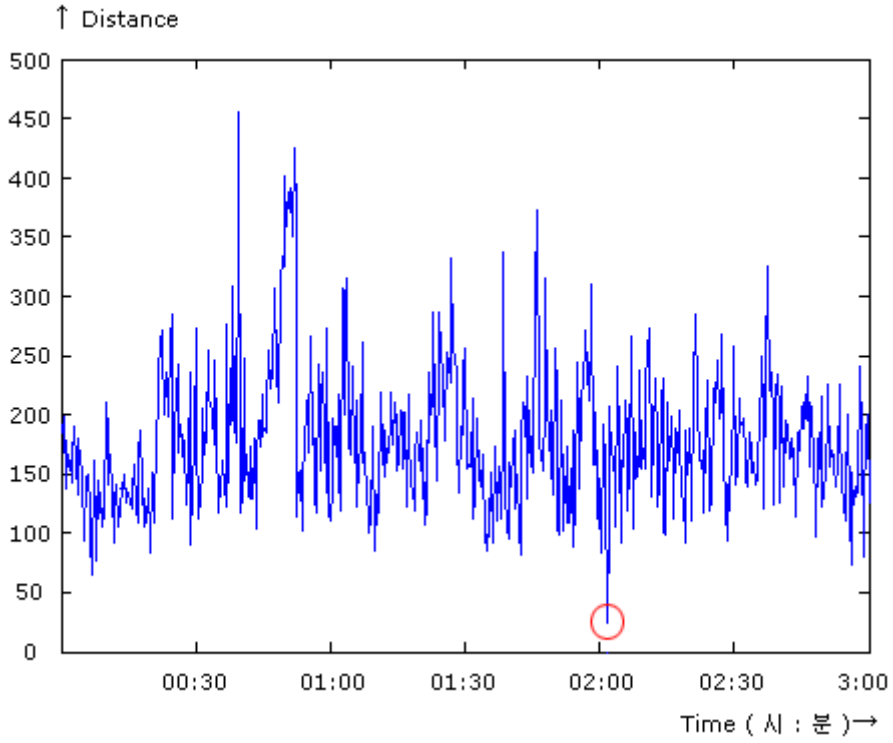

그림 4-1. 다중 특징 벡터의 히스토그램을 이용한 유사도 측정

Fig. 4-1 The similarity measurement using the histogram of multiple feature vectors.

4.2 검색 속도

시간 순서 오디오 검색에서는 30 분 짜리 오디오에서 원하는 위치를 찾을 때 1 분 35 초라는 시간이 소요되었다. 왜냐하면 각각의 프레임마다 히스토그램을 구하여 유사도를 측정해가기 때문에 많은 시간이 소요된다. 하지만 본 논문에서 제안한 알고리즘에서는 전처리를 통하여 단지 기준 신호에서 나오는 특징 벡터와 히스토그램만을 추출하기 때문에 검색 시간이 매우 줄어들게 된다. 그리고 같은 코드를 먼저 검색하고 난 후에 그 코드에 맞는 프레임만 찾아서 히스토그램을 검색하기 때문에 검색 속도에 많은 향상을 가져온다. 그림 4-1 은 3 시간짜리 오디오의 모든 프레임을 다 검색한 그림이다. 여기서는 총 540 번의 유사도가 측정이 되었다. 그림 4-1 은 같은 코드를 가진 프레임만을 추출하여 유사도를 구한 그림이다. 여기서는 단지 120 개의 프레임만으로 유사도를 측정하여 원하는 위치의 곡을 파악할 수 있었다.

기존 논문에서 upper bound proof 를 이용하여 검색하였을 때는 30 분 짜리 오디오에서만 1 분 35 초라는 시간이 소요되었지만 본 논문에서 제안한 알고리즘은 3 시간의 오디오 stream 에서 원하는 위치의 곡을 찾는데 단지 9 초라는 시간만 걸려서 속도 면에서 대략 50 배 정도 향상되었다는 것을 알 수 잇다.

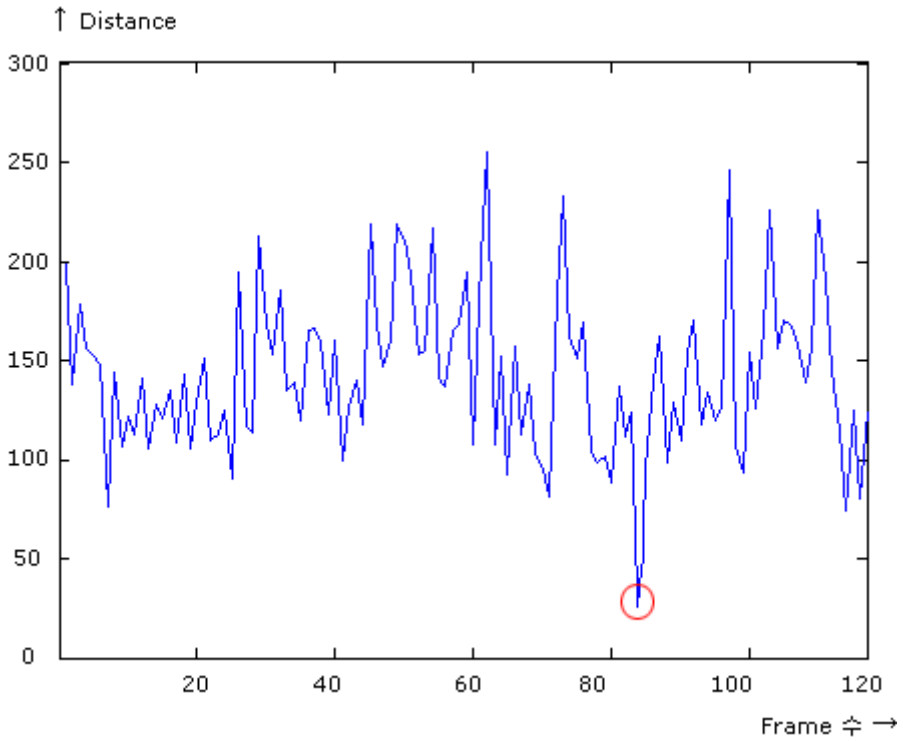

그림 4-2. 제안한 고속 오디오 검색의 결과

Fig. 4-2 The result of the proposed quick audio search.

제 5 장 결 론

본 논문에서는 기존의 오디오 검색에서 정확성과 검색 속도 문제를 개선하여 높은 정확도를 가진 실시간 검색을 구축하기 위한 고속 검색 알고리즘을 설계하였다. 기존의 논문에서는 ZCR 과 같은 단일 특징 벡터를 이용하여 검색 속도를 개선하였지만, 본 논문에서는 정확도를 높이기 위하여 다중 특징 벡터를 이용하였다. 어떤 장르에도 구애 받지 않고 정확한 검색을 할 수 있는 다중 특징 벡터의 조합을 실험을 하였고 각각의 특징 벡터의 조합에 따른 검색 속도를 측정하였다. 그 결과, ZCR 의 단일 특징 벡터로 검색하였을 때 보다 더 정확하고 더 빠른 검색을 할 수 있는 다중 특징 벡터의 조합을 구성할 수 있었다. 그리고 여러 가지 유사도 측정 방법을 비교하여 성능이 가장 좋은 방법을 택하여 알고리즘을 구성하였다.

제안한 검색의 실험 결과, 다중 특징 벡터의 구성으로 검색의 정확도에서 우수한 특성을 보였으며, 전처리 단계를 통하여 기존의 검색 방법 보다 속도가 약 50 배정도 빠른 고속 검색을 할 수 있었다. 많은 오디오의 특징 벡터들을 히스토그램으로 나타내어서 데이터 베이스의 양을 대폭 줄 일 수 있으며, 검색을 하기 위해서는 먼저 같은 코드를 찾아 동일한 코드를 가진 프레임만을 검색하여 유사도를 측정하기 때문에 시간 순서 오디오 검색 보다는 매우 탁월하게 검색 속도가 향상되었다. 디지털 방송 서비스가 활성화 되면서 더더욱 멀티미디어의 양은 많아지고 그 많은 멀티미디어

속에서 우리가 원하는 정보를 검색하기 위해서는 내용 기반 검색이 무엇보다 중요하다. 그러므로 더 많은 특징 벡터들의 분석을 통하여 장르뿐만 아니라, 목소리에 따른 분류, 악기에 따른 분류, 멜로디에 따른 분류 등을 검색할 수 있는 특징 벡터들의 구성이 연구되어져야 할 것이다.

참 고 문 헌

[1] Taegu Internet Broadcasting System , http://www.tibs.co.kr/

[2] G. Smith, H. Murase, and K. Kashino, “Quick audio retrieval using active search,” *Proc of ICASSP’98*, vol. 6, pp.3777-3780, 1998.

[3] K. Kashino, G. Smith, and H.Murase, “Time-series active search for quick retrieval of audio and video,” *Proc of ICASSP99*, vol.6, pp. 2993~2996, March 1999.

[4] G. Tzanetakis, Manipulation, Analysis and retrieval systems for audio, Ph.D dissertation, June 2002.

[5] J. Jose Burred and A. Learch, “Hierachical automatic audio signal classification,” *Proc of J. Audio Eng. Sec.*, vol.52, pp. 724-739, July/August 2005.

[6] A. Kimura, K. Kashino, T. Kurozumi, and H. Murase, “Very quick audio searching : introducing global pruning to the time-series active search,” *Proc of IEEE*, pp. 1429-1432, 2001.

[7] 반지혜, 김기만, 박규식, “ 다중 특징 벡터를 이용한 고속 오디오 검색,” 한국음향학회 학술 발표대회 논문집, 제 1(s)호, pp. 351-354, May 2004.

[8] J. T. Foote, “Content-based retrieval of music and audio,” *Proc of SPIE*, vol.3229, pp.138-147, 1997.

[9] J. D. Hoyt and H. Wechsler, “Detection of human speech in structured noise,” *Proc of ICASSP’94*, vol.2, pp. 237-240, 1994.

[10] K. Kahino, “A quick search method for audio and video signals based on histogram pruning,” *Proc of IEEE*, vol.5, no.3, pp. 348-357, September 2003.

[11] E. wold, T. Blum, D. Keislar, and J. Wheaton, “Content-based classification, search, and retrieval of audio,” *Proc of IEEE Multimedia*, vol.3, no.3, pp.27-36, 1996.

[12] T. Zhang and J. C. Kuo, “Hierarchical system for content-based audio classification and retrieval,” *Proc of SPIE*, vol. 3527, pp. 398-409, 1998.

[13] Lu Guojun, “Indexing and Retrieval of Audio: A survey,” Journal of Multimedia Tools and Applications, pp. 269-290, 2001.

[14] J. T. Foote, “An overview of Audio Information Retrieval,” *Proc of ACM Multimedia*, pp. 2-10, 1998.

[15] S. J. Young, M. G. Brown, J. T. Foote, G. J. F. Jones, and K.S. Jones, “Acoustic indexing for multimedia retrieval and browsing,” *Proc of ICASSSP’97*, vol. 1, pp. 199-202, 1997.

[16] P. Y. Rolland, G. Raskins, J. Ganascia, “Musical content-based retrieval : an overview of the melodiscov approach and system,” *Proc of ACM Multimedia*, pp.81-84, 1999.

[17] J. J. Aucouturier and F. Pachet, “Representing musical genre : A state of the art,” *Proc of J. New Music Research*, vol. 32, no.1, 2003.

[18] G. Tzanetakis, G. Essl, and P. Cook, “Automatic musical genre classification of audio signals,” *Proc of 2nd Ann. ISMIR*, 2001.

[19] R. S. Boyer and J. S. Moore, “A fasr string matching algorithm,” *Proc of Commun. ACM*, vol. 20, no. 10, pp. 702-772, 1997.

[20] J. H. Knuth, J. H. Morrism, and V. R. Pratt, “Fast pattern matching in strings,” *Proc of SIAN J. Comput*, vol. 6, no. 2, pp. 323-350, 1997.

감사의 글

대학원에 입학하여 짧은 시간이지만 2년 동안 공부한 내용을 하나의 논문으로 만들었습니다. 대학원 생활 동안 저에게 많은 가르침을 주시고 항상 따뜻하게 대해 주셨던 저의 우상이신 김기만 교수님께 진심으로 감사 드립니다. 그리고 학부 생활부터 지도해주시고 지켜 봐주신 김동일 교수님, 조형래 교수님, 정지원 교수님, 민경식 교수님과 비록 학부생활 때 뵙지 못했지만 항상 격려해주신 윤영 교수님께도 감사 드립니다. 대학원 진학부터 지금까지 저의 상담자인 윤준이 오빠, 그리고 함께 실험실 생활을 했던 승용 큰 형님, 외형이 오빠, 영근이 오빠, 진석이 오빠, 형준이, 세영이 오빠, 재국이 오빠에게도 감사의 말을 전합니다. 그리고 항상 어려울 때 저에게 늘 힘이 되어 주었던 규호, 보영이, 주인이, 오자매 그리고 많은 99학번 동기들께도 고맙다는 말을 전합니다.

마지막으로 저의 대학원 진학을 전폭적으로 밀어주시고 저를 믿어주셨던 사랑하는 부모님과 사랑스런 동생 지은이와 저에게 많은 도움을 주셨던 친지 분들께도 정말 진심으로 감사 드리고 지금은 하늘나라에 계신 할아버지께 정말 감사하다는 말을 전하고 싶습니다.

많은 사람들께서 베풀어 주신 사랑과 관심으로 제가 이렇게 성장할 수 있었던 것 같습니다. 그 은혜 절대로 잊지 않고 저 또한 다른 사람들에게 많은 사랑을 베풀 수 있고 도움이 될 수 있는 사람이 되겠습니다.