정규논문 (Regular Paper)
방송공학회논문지 제17권 제4호, 2012년 7월 (JBE Vol. 17, No. 4, July 2012)
http://dx.doi.org/10.5909/JBE.2012.17.4.659

이진 핑거프린트의 결합에 의한 강인한 오디오 핑거프린트

장 달 원[a)], 이 석 필[a)‡]

Audio Fingerprint Based on Combining Binary Fingerprints

Dalwon Jang[a)] and Seok-Pil Lee[a)‡]

요 약

이 논문에서는 이진 핑거프린트의 결합을 이용해 새로운 이진 오디오 핑거프린트를 만드는 방법을 제안하다. 필립스 핑거프린팅 시스템을 활용하여, 그 시스템에서 활용한과 비슷한 특성을 가질 것이라 예상되는 기본 이진 핑거프린트를 여러 개 추출하고, 기본 이진 핑거프린트들의 투표로 하나의 이진 오디오 핑거프린트를 결정한다. 정합단에서는 이진 핑거프린트를 이용하는 것이 아니라, 기본 이진 핑거프린트들의 합을 이용하여 거리를 계산한다. 실험을 통해서 제안하는 방법으로 만들어진 핑거프린트가 그것의 기초가 되는 기본 이진 핑거프린트들보다 향상된 성능을 보임을 확인할 수 있었다. 이 방법을 이용해서 기존의 이진 핑거프린트의 성능을 강화하거나 새로운 이진 핑거프린트를 만들 수 있을 것이라 기대된다.

Abstract

This paper proposes the method to extract a binary audio fingerprint by combining several base binary fingerprints. Based on majority voting of base fingerprints, which are designed by mimicking the fingerprint used in Philips fingerprinting system, the proposed fingerprint is determined. In the matching part, the base fingerprints are extracted from the query, and distance is computed using the sum of them. In the experiments, the proposed fingerprint outperforms the base binary fingerprints. The method can be used for enhancing the existing binary fingerprint or for designing a new fingerprint.

Keyword : audio identification, audio fingerprinting, fingerprint matching, fingerprint combination, binary fingerprint

Ⅰ. 서 론

오디오 인식 기술, 주로 오디오 핑거프린팅 (audio fingerprinting)으로 불리는 기술은 지난 10년간 다양한 기술들이 제안되었다[1-12]. 핑거프린팅 시스템은 핑거프린트 (fingerprint) 라고 불리는 짧은 길이의 벡터를 입력 오디오 콘텐츠로부터 추출하고, 데이터베이스에 저장된 핑거프린트들과의 거리 계산을 통해서 입력된 오디오 콘텐츠의 정보를 추출하는 시스템이다. 과거, 관련 연구에서는 핑거프린팅 시스템을 제안하거나 강인한 핑거프린트를 제안하였다[2-10]. 또, 기존의 핑거프린팅 시스템에서 거리 계산을 수행하는 방법에 집중하여, 거리 계산 방식을 개선하여 핑거프린팅 시스템의 성능을 향상시키기도 하였고[11-12], 핑거프

a) 전자부품연구원 디지털 미디어 연구센터 (Digital Media Research Center, KETI)
‡ 교신저자 : 이석필 (Seok-Pil Lee)
E-mail: lspbio@keti.re.kr
Tel: +82-2-6388-6640-, Fax: +82-2-6388-6659
· 접수일 (2012년5월11일),수정일(2012년6월26일),게재확정일(2012년7월12일)

린트를 이용해서 오디오 신호의 특성을 분석하려는 시도도 있어왔다[13].

이 논문에서는 이진 핑거프린트의 결합을 이용해서 만들어진 새로운 이진 오디오 핑거프린트를 만들고 그것의 거리를 계산하는 방법을 제안한다. 유사한 특성을 가지는 기본 이진 핑거프린트를 다수 개 만든 후에 그것들에 대해서 투표를 수행하여 최종 이진 핑거프린트를 만들고, 거리 계산 시에는 기본 이진 핑거프린트의 합을 이용해서 가중치를 주는 방법을 사용한다. 이 방법으로 인해서, 이진 핑거프린트의 길이를 유지하면서 기존 핑거프린트의 성능을 강화할 수 있으며, 또한 기존의 핑거프린트들을 활용해서 새로운 핑거프린트를 만들 수도 있다. 이 논문에서는 핑거프린팅 분야에서 많이 연구되고 있는 필립스 핑거프린팅 시스템[2]에 기초하여 필립스 핑거프린팅 시스템의 핑거프린트를 하나의 기본 이진 핑거프린트로 두었다. 그리고 그와 유사한 형태의 기본 이진 핑거프린트들을 만들고 최종 핑거프린트를 추출해서 성능을 향상시켰다. 논문의 주요 기여는 여러 개의 이진 핑거프린트를 결합하여 각각의 성능보다 더 나은 성능을 낼 수 있다는 점이다. 논문에서 제안하는 방법은 기존의 이진 핑거프린트의 성능을 강화하기 위해서, 또는 기존의 이진 핑거프린트를 활용하여 새로운 이진 핑거프린트를 만들기 위해서 사용할 수 있다.

논문의 나머지 부분은 다음과 같이 채워진다. II장에서는 관련 연구에 대해서 설명한다. III장에서는 핑거프린트 추출 방법, IV장에서는 핑거프린트 사이의 거리 계산 방법을 설명하고, V장에서는 다양한 실험 결과를 보인다. VI장에서 제안한 핑거프린트의 장단점을 분석하고, 앞으로 보완해야 할 부분을 밝힌다. VII 장에서 논문의 결론을 적는다.

II. 관련 연구

1. 핑거프린팅 시스템

이 절에서는 핑거프린트 추출의 기본 형태와 핑거프린팅 시스템에 대해서 설명할 것이다. 그림 1.에 오디오 신호로부터 핑거프린트를 추출하는 시스템의 기본 구조가 그려져 있다. 그림에서 보는 것과 같이 정규화, 변환, 계산, 가공 과정이 존재한다. 입력 오디오에 대해서 다양한 기본적 왜곡에 대해서 정규화를 시켜주는 과정이 첫 번째 시행된다. 시스템에서 정한 특정 주파수의 신호로 다운샘플링하고, 스테레오 신호는 모노 신호로 변경하고, 프레임별로 나누는 과정이 존재한다. 이 과정 후에는 프레임별로 푸리에 변환, 웨이블렛 변환 등의 변환 과정을 거치고, 밴드를 나누어 에너지나 중심값을 계산하는 등의 과정이 존재한다. 다음으로는 이러한 계산 결과들을 핑거프린트로 가공하는 과정이 존재한다. 문턱값을 사용하여 이진화를 수행하는 과정 등이 이 과정에 포함된다.

핑거프린팅 시스템의 기본 구조는 아래의 그림 2.와 같다. 이것은 오디오 핑거프린팅 시스템, 비디오 핑거프린팅 시스템에 관계없이 동일한 구조를 가지며, 여기서는 오디오 핑거프린팅 시스템을 기준으로 설명할 것이다. 이 시스템을 위해서는 먼저 핑거프린트 추출 방법과 핑거프린트 간의 거리 계산 방법이 정의되어 있어야 할 것이다. 콘텐츠 데이터베이스를 먼저 모으고, 그것에서부터 핑거프린트를 추출한 핑거프린트 데이터베이스를 생성한다. 핑거프린트 데이터베이스는 콘텐츠의 핑거프린트는 물론, 그것에 상응하는 콘텐츠의 정보를 가지고 있다. 핑거프린트 데이터베이스를 가지고 있다는 가정 하에 시스템이 구성된다. 쿼리(query) 입력으로 짧은 길이의 오디오가 들어오면 그것에

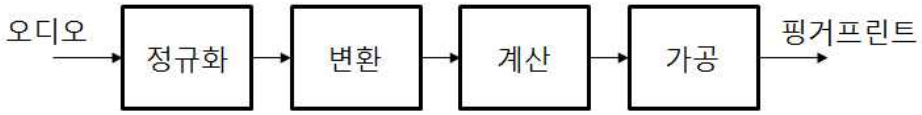

그림 1. 핑거프린트 추출의 기본 구조
Fig. 1. Basic structure of fingerprint extraction

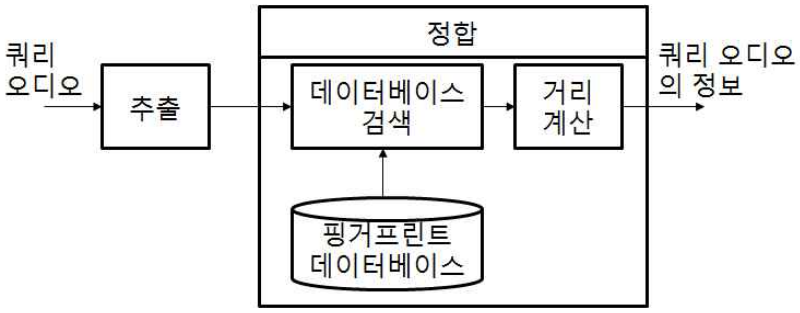

그림 2. 핑거프린팅 시스템의 일반적 구조
Fig. 2. Common structure of fingerprinting system

대해서 핑거프린트를 추출한다. 추출한 정합 (matching) 과정을 거치고, 정합 과정에서 핑거프린트는 핑거프린팅 시스템의 결과물인 입력에 관한 정보가 출력된다. 정합 과정은 크게 두 과정을 나눠진다. 데이터베이스 검색 과정과 거리 계산 과정이다. 데이터베이스 검색 과정을 데이터베이스에 있는 수많은 핑거프린트들로부터 쿼리 오디오의 핑거프린트와 비슷할 것이라 예상되는 부분집합을 추려내는 과정이다. 추려낸 부분집합에 존재하는 모든 핑거프린트들과 쿼리 오디오의 핑거프린트들 사이의 거리를 계산해서 최소의 거리를 가지는 핑거프린트의 정보를 출력으로 낸다. 최소값을 찾기 위해서 거리 계산 방법은 미리 정의되어 있어야 하며, 핑거프린팅 시스템의 인식 성능을 핑거프린트의 특징과 거리 계산 방법이 좌우하며, 데이터베이스 검색 방법은 핑거프린트 시스템의 시간적 성능을 높이는 게 목적이다.

2. 필립스 핑거프린팅 시스템의 핑거프린트 추출

이 절에서는 필립스 핑거프린팅 시스템 [2]의 핑거프린트 추출에 대해서 설명한다.

오디오 입력이 들어오면 모노 신호로 변환 후 약 5.5kHz[1] 로 다운샘플링한다. 이 신호를 2048 샘플 (≈ 0.37s) 길이를 가지는 윈도우를 사용하여 프레임을 나누고, 이 프레임은 31/32만큼 오버랩하면서 11.6ms씩 이동한다. 각 프레임의 신호를 2048포인트 푸리에 변화하여 스펙트럼을 구한 후, 300Hz에서 2000Hz 사이의 신호를 33개의 부밴드 (sub-band) 로 나눈다. 그리고 각 부밴드의 에너지를 계산한 후, 시간상, 공간상으로 이웃하는 밴드의 에너지들끼리 차를 구하고 그로부터 하나의 비트를 계산해 낸다. n번째 프레임의 m번째 부밴드의 에너지를 $E(n,m)$이라고 했을 때 n번째 프레임의 m번째 비트 $F(n,m)$은 식 (1)과 같이 정해진다.

이 중, $E(n,m)-E(n,m+1)-(E(n-1,m)-(E(n-1,m+1))$, 이 계산 부분에 대해서 [9]와 [10]에서는 필터링으로 해석했지만, 이것은 ((두 개 값의 산술 평균) - (두 개 값의 산술 평균))$\times 2$ 이라고 생각할 수 있다. 이 논문에서는 이 점을 활용하여 기본 핑거프린트 추출에 이용한다.

III. 핑거프린트 추출

이 장에서는 논문에서 제안하는 핑거프린트를 추출하는 방법에 대해서 설명한다. 핑거프린트 추출 과정을 요약한 과정이 아래의 그림 3.에 요약되어 있다. 그림에서 보는 바와 같이 제안하는 핑거프린트는 L개의 기본 이진 핑거프린트를 추출한 후에 이것들로부터 하나의 이진 핑거프린트를

$$F(n,m)=\begin{cases}1, & \text{if } E(n,m)-E(n,m+1)-(E(n-1,m)-(E(n-1,m+1))>0\\ 0, & \text{if } E(n,m)-E(n,m+1)-(E(n-1,m)-(E(n-1,m+1))\le 0\end{cases} \quad (1)$$

1) 44.1kHz/8 = 5.5125kHz

만든다. 자세한 사항을 각 절에서 설명될 것이다.

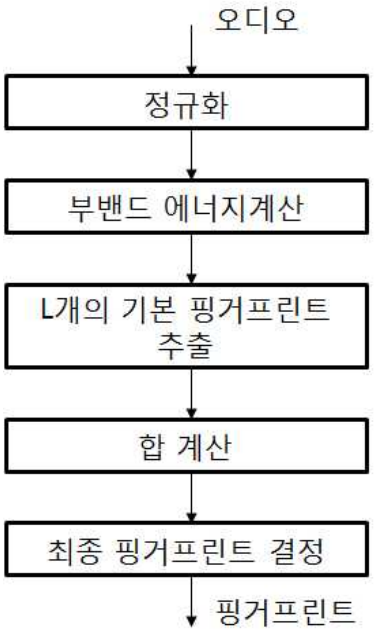

그림 3. 핑거프린트 추출 과정
Fig. 3. Fingerprint extraction

1. 기본 이진 핑거프린트 추출

오디오 입력에 대해서 전처리 과정을 거치고 각 프레임마다 푸리어 트랜스폼을 하고, 부밴드를 나누고, 부밴드의 에너지를 계산한다. 이 과정은 II장에서 설명한 필립스 핑거프린팅 시스템의 과정을 그대로 사용한다. 오디오 신호를 모노로 바꾸고, 5.5 khz로 다운샘플링한다. 2048포인트 길이를 가지고, 11.6ms씩 이동하는 프레임 하나에서 33개의 부밴드 에너지를 계산한다. 부밴드의 에너지값을 이용해서 L개의 기본 핑거프린트를 추출한다. 앞장과 마찬가지로 n번째 프레임의 m번째 부밴드의 에너지를 $E(n,m)$이라 나타내고, 표현의 편의를 위해서 $E_a = E(n,m)$, $E_b = E(n,m+1)$, $E_c = E(n-1,m)$, $E_d = E(n-1,m+1)$로 나타낸다. 각각의 기본 핑거프린트를 $\overrightarrow{F_{B1}}$, $\overrightarrow{F_{B2}}$, …, $\overrightarrow{F_{BL}}$으로 표현한다. 우리의 시스템에서는 $L=5$로 두고 아래의 다섯 가지 기본 핑거프린트를 추출하였다.

$\overrightarrow{F_{B1}}$: 산술 평균을 이용

$$\overrightarrow{F_{B1}}(n,m) = \begin{cases} 1, & \text{if}\,(E_a+E_d)-(E_b+E_c) > 0 \\ 0, & \text{if}\,(E_a+E_d)-(E_b+E_c) \le 0 \end{cases} \quad (2)$$

$\overrightarrow{F_{B2}}$: 기하 평균을 이용

$$\overrightarrow{F_{B2}}(n,m) = \begin{cases} 1, & \text{if}\,(E_a\times E_d)-(E_b\times E_c) > 0 \\ 0, & \text{if}\,(E_a\times E_d)-(E_b\times E_c) \le 0 \end{cases} \quad (3)$$

$\overrightarrow{F_{B3}}$: 조화 평균을 이용

$$\overrightarrow{F_{B3}}(n,m) = \begin{cases} 1, & \text{if}\,E_aE_b/(E_a+E_d)-E_bE_c/(E_b+E_c) > 0 \\ 0, & \text{if}\,E_aE_b/(E_a+E_d)-E_bE_c/(E_b+E_c) \le 0 \end{cases} \quad (4)$$

$\overrightarrow{F_{B4}}$: 제곱근 계산 후 산술 평균을 이용

$$\overrightarrow{F_{B4}}(n,m) = \begin{cases} 1, & \text{if}\,(\sqrt{E_a}+\sqrt{E_d})-(\sqrt{E_b}+\sqrt{E_c}) > 0 \\ 0, & \text{if}\,(\sqrt{E_a}+\sqrt{E_d})-(\sqrt{E_b}+\sqrt{E_c}) \le 0 \end{cases} \quad (5)$$

$\overrightarrow{F_{B5}}$: 자연로그 계산 후 기하 평균을 이용

$$\overrightarrow{F_{B5}}(n,m) = \begin{cases} 1, & \text{if}\,(log(E_a)\times log(E_d))-(log(E_b)\times log(E_c)) > 0 \\ 0, & \text{if}\,(log(E_a)\times log(E_d))-(log(E_b)\times log(E_c)) \le 0 \end{cases} \quad (6)$$

열거한 것 중, $\overrightarrow{F_{B1}}$은 기존의 필립스 핑거프린트와 동일하다. 나머지 기본 핑거프린트들도 $\overrightarrow{F_{B1}}$의 변형이라고 할 수 있다. 여기에서 알 수 있듯이, 이 논문에서 제안하는 핑거프린트는 특정한 계산을 찾은 것이 아니라, $\overrightarrow{F_{B1}}$와 비슷한 특성을 보일 수 있도록 계산과 가공 과정을 만들어서 사용한 것이다. 이 핑거프린트들은 서로 유사한 특성을 나타낼 것이며, 시행 결과, 모두 비슷한 수준의 성능을 보임을 확인할 수 있었다.

여기까지의 추출과정이 그림 1.에서 나타낸 핑거프린트 추출 과정을 모두 거친 것으로, 각각의 기본 핑거프린트는

일반적인 핑거프린팅 시스템에 사용할 수 있다. 제안하는 핑거프린트는 그림 1.에 나타낸 일반적 과정 외에 그림 3.에 보여지는 아래의 두 과정을 더 거치게 된다.

2. 합 계산

추출한 기본 핑거프린트는 한 프레임당 32비트 길이의 신호로 비트 0, 비트 1의 값을 가지고 있다. 이 비트 값을 0과 1의 실수값으로 가정하고, 그것들의 비트 위치별 합을 계산한다. 5개의기본 핑거프린트를 더하면 $\{0,1,2,\cdots,L\}$의 값을 갖는 길이 32의 벡터가 된다.

3. 최종 핑거프린트 결정

합 계산한 결과를 문턱값과 비교해서 최종 핑거프린트의 비트 값을 결정한다. 우리는 $L=5$라 두었기 때문에, $\{0,1,2,3,4,5\}$ 총 여섯 가지의 값이 합 계산 결과 나올 수 있다. 여기서 문턱값을 2.5 ($=\frac{L}{2}$) 로 두고, 이보다 작으면 비트 0, 크면 비트 1을 할당한다. 즉, $\{0,1,2\}$의 경우에는 비트 0를, $\{3,4,5\}$의 경우에는 비트 1을 할당한다. 합을 계산한 값을 각 비트 위치에서 비트 1의 개수라고 할 수 있고, 문턱값 2.5를 사용해 두 가지 경우로 나눈 것은 비트 0과 비트 1에 대해서 다수를 이루는 비트의 값을 따르는 것이라 할 수 있다. 즉, L개의 이진 핑거프린트로부터 다수 투표 (majority voting) 방법을 통해서 최종 핑거프린트를 결정하는 것과 같다. 표 1.에 핑거프린트를 결정하는 예가 나와 있다.

표 1. 최종 핑거프린트 결정의 예
Table 1. Example: determination of fingerprint

$\overrightarrow{F_{B1}}$	$\overrightarrow{F_{B2}}$	$\overrightarrow{F_{B3}}$	$\overrightarrow{F_{B4}}$	$\overrightarrow{F_{B5}}$	합	최종 핑거프린트
1	1	1	1	1	5	1
1	1	1	0	1	4	1
0	0	0	0	0	0	0
1	0	0	1	1	3	1
0	0	0	1	1	2	0
1	0	0	0	0	1	0

다수 개의 기본 핑거프린트를 추출하였지만, 데이터베이스에는 최종 핑거프린트만이 저장된다. 최종 핑거프린트의 길이는 각각의 핑거프린트의 길이와 같기 때문에 논문에서 제안하는 핑거프린트는 [2]의 핑거프린팅 시스템과 같은 크기의 핑거프린트 데이터베이스는 가지게 될 것이다. 여기서는 한 프레임 당 32비트의 핑거프린트를 추출하여 저장한다.

IV. 핑거프린트 거리계산

II장에서 설명한 바와 같이 핑거프린트를 두 개의 핑거프린트 사이의 거리를 계산하여 콘텐츠를 인식한다. 같은 핑거프린트라도 거리를 계산하는 방식에 따라서 다른 성능을 보이게 된다. 이 장에서는 논문에서 제안하는 핑거프린트의 거리 계산 방법에 대해서 설명한다.

그림 4.는 핑거프린트 사이의 거리를 계산하는 방법을 설명하고 있다. 앞 장에서 설명한 방법대로 추출된 핑거프린트들이 데이터베이스에 저장되어 있다고 가정하고, 쿼리 오디오가 입력되었을 경우에 쿼리 오디오에서 기본 핑거프린트를 추출하고 합을 계산한다. 합을 계산한 벡터와 데이

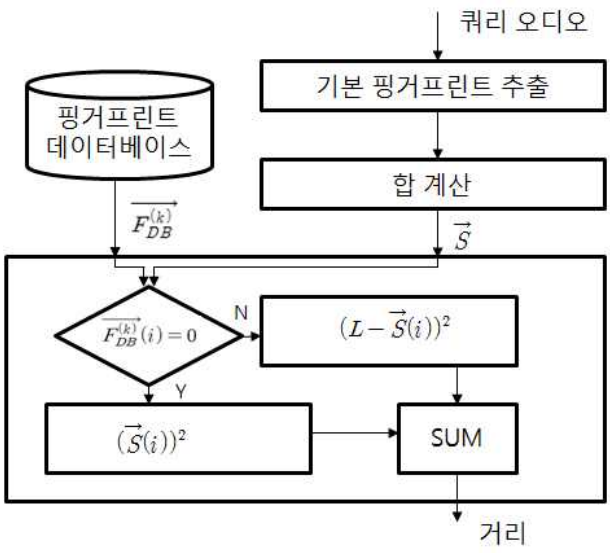

그림 4. 핑거프린트 간 거리 계산 과정
Fig. 4. Distance computation between fingerprints

터베이스에 미리 저장되어 있던 핑거프린트를 이용해서 거리를 구한다. 총 K개의 핑거프린트가 데이터베이스에 미리 저장되어 있다고 가정하고, 그 중 하나의 핑거프린트를 $\overrightarrow{F_{DB}^{(k)}}$ $(k=1,2,\cdots,K)$라 칭한다. 정합과정에서는 데이터베이스에 저장된 K개의 핑거프린트 중 최소거리를 가지는 핑거프린트를 찾아야 할 것이다.

1. 기본 핑거프린트 추출

기본 핑거프린트를 추출하는 방법은 앞 장에서 설명한 것과 같은 과정을 거친다. 입력 쿼리에 대해서 전처리 과정을 거치고 L개의 기본 핑거프린트를 추출한다. 우리는, 쿼리 오디오의 입력에서 추출할 수 있는 프레임을 128로 가정하고 총 32bit/프레임 * 128 프레임 = 4096 bit의 기본 핑거프린트들을 각각 추출해 낸다. 설명의 편의를 위해서, 이 장에서 설명되는 벡터는 4096bit의 벡터를 하나의 핑거프린트 단위로 본다.

2. 합 계산

기본 핑거프린트로부터 합을 계산한다. 합 계산의 방식은 전 장에서 설명한 건과 같다. 합을 계산한 4096의 길이를 가지는 벡터는 $\vec{S}$ 라고 정의하고, $\vec{S}(i)$를 i번째 원소 $(i=1,2,\cdots,4096)$ 라고 정의한다.

3. 거리 계산

데이터베이스에 저장된 핑거프린트 $\overrightarrow{F_{DB}^{(k)}}$와 합 벡터 $\vec{S}$ 사이의 거리는 다음과 같이 정의한다.

$$d(\vec{S},\overrightarrow{F_{DB}^{(k)}}) = \sum_{i=1,F_{DB}^{(k)}(i)=0}^{4096} (\vec{S}(i))^2 + \sum_{i=1,F_{DB}^{(k)}(i)=1}^{4096} (L-\vec{S}(i))^2 \quad (8)$$

데이터베이스에 저장된 핑거프린트의 $\overrightarrow{F_{DB}^{(k)}}$의 i번째 비트 값이 0인 경우에 대해서 $\vec{S}(i)$=0와 $\overrightarrow{F_{DB}^{(k)}}$의 비트 값이 1인 경우에 대해서 $\vec{S}(i)=L$이 가장 이상적으로 정합되는 것이다. 그에 미치지 못 할 경우에는 차이값에 대해서 제곱의 가중치를 두고 전체 합을 계산한다.

표 2. 거리 계산의 예
Table 2. Example: distance computation

bit index	$\overrightarrow{F_{DB}^{(k)}}$	$\overrightarrow{F_{B1}}$	$\overrightarrow{F_{B2}}$	$\overrightarrow{F_{B3}}$	$\overrightarrow{F_{B4}}$	$\overrightarrow{F_{B5}}$	$\vec{S}$	거리
1	1	1	1	1	1	1	5	0
2	0	1	1	1	0	1	4	16
3	1	0	0	0	0	0	0	25
4	1	1	0	1	1	1	4	1
5	1	0	1	1	1	0	3	4
6	1	1	0	0	1	0	2	9
7	1	0	0	1	0	0	1	16
8	0	0	0	0	0	0	0	0
9	0	0	1	1	1	0	3	9
…	…	…	…	…	…	…	…	…
4096	0	1	1	1	1	1	5	25

표 2.에 이해를 돕기 위한 예가 있다. 총 4096 비트에 대해서, 쿼리 오디오에서 $\overrightarrow{F_{B1}}$, $\overrightarrow{F_{B2}}$, $\overrightarrow{F_{B3}}$, $\overrightarrow{F_{B4}}$, $\overrightarrow{F_{B5}}$, 다섯 가지 종류의 기본 핑거프린트를 추출하였고 그것에 기반하여 $\vec{S}$를 계산하였다. 그리고 데이터베이스에서 임의의 핑거프린트 $\overrightarrow{F_{DB}^{(k)}}$ 가져왔을 때의 거리이다. $\overrightarrow{F_{DB}^{(k)}}$의 비트가 1일 때는 $\vec{S}$의 값 5, 4, 3, 2, 1, 0 에 대해서 거리는 0, 1, 4, 9, 16, 25의 값을 가지게 되고, 반대로 비트가 0일 때는 $\vec{S}$의 값이 5, 4, 3, 2, 1, 0에 대해서 25, 16, 9. 4, 1, 0의 값을 가지게 된다. 그리고 최종적으로 4096비트에서 계산된 거리를 다 더하면 최종 거리를 얻을 수 있다.

V. 실험결과

1. 실험 셋팅

본 논문에서 제안한 핑거프린트는 필립스 핑거프린팅 시

스템에 제안된 핑거프린트와의 비교를 통하여 성능을 보여준다. 앞에서 설명한 바와 같이 제안한 시스템의 전처리 단계는 필립스 핑거프린팅 시스템의 그것에 따른다. 논문 [2]에서는 3초 가량의 입력을 가정하고 256 프레임의 핑거프린트를 모아서 총 8192bit에서 식별 결과를 보여주었지만, 우리 시스템에서는 2초 이하의 입력을 가정하였다. 이 논문에서는 입력 길이를 바꾸는 것이 전체 성능에 영향을 미치는 것은 사실이나, 두 핑거프린트의 성능이 뒤바뀌지는 않는다는 가정 하에, 2초 이하의 입력에서 핑거프린트의 성능을 비교한다. 총 128 프레임을 이용하였으며 실제 길이는 약 1.8초 정도의 입력을 이용한다.

핑거프린트의 성능을 보이기 위해서 false positive (FP) rate 와 false negative (FN) rate 를 도시한 receiver operating characteristic (ROC) 커브를 보인다. 하나의 문턱값에 대해서, FP rate는 서로 다른 오디오 클립의 쌍에 대해서 같다고 판단내리는 비율이며, FN rate는 서로 같은 오디오 클립의 쌍에 대해서 다르다고 판단내리는 비율이다. ROC 커브를 도시하는 방법은 핑거프린트 자체의 성능을 나타내기 위해서 많이 사용되어 왔다[8,11]. 두 가지의 실험 결과를 얻기 위해서 약 25,000 쌍의 서로 같은 오디오 클립 쌍과 약 1억 5천만 쌍의 서로 다른 오디오 클립 쌍을 이용하였다. 왜곡을 거치지 않은 원본 오디오와 그에 상응하는 왜곡을 가한 오디오를 만들어서 서로 같은 쌍을 만들었다. 왜곡을 거치치 않은 원본 오디오와 그와 전혀 다른 부분의 오디오에 왜곡을 가한 오디오를 이용해서 서로 다른 쌍을 만들었다. 왜곡을 거친 오디오가 핑거프린팅 시스템의 쿼리 입력으로 들어오고, 원본 오디오가 데이터베이스에 저장되어 있다는 가정 하에 쌍이 만들어진 것이다. 위의 쌍들은 총 200곡의 가요를 이용해서 만들었다.

다양한 왜곡을 가정하고, 왜곡된 오디오를 생성하여 실험에 사용하였다[8]. 실험에 사용된 왜곡들은 다음의 리스트와 같다.

1) Time delay (TD): 5.8ms 딜레이
2) Octave band equalization (EQ1): 옥타브밴드로 나누고, 밴드 순서대로 -6dB와 6dB를 번갈아가면서 주는 등화 과정
3) Volume change (V): 볼륨을 계속 변동
4) Echo (E): 필터를 통해서 오래된 라디오와 같은 효과를 줌
5) Bandpass filter (BPF): 400Hz-4kHz 밴드 패스 필터
6) WMA encoding (WMA): 64kb/s WMA 인코딩
7) 1/3 octave band equalization (EQ2): 30 밴드 팝 형태 등화

모든 왜곡 후에는 96kbps MP3 encoding (MP3) 이 따랐으며, 이 실험셋에서 왜곡된 오디오는 쿨에디트 프로그램을 통해서 생성되었다. 각각의 왜곡을 연속적으로 가한 복합 왜곡을 통해서 강한 왜곡에 대해서도 성능을 평가하였다. 다음에 열거한 복합 왜곡을 실험에 사용하였다.

8) EQ2 + WMA + MP3
9) EQ1 + V + BPF + WMA + MP3

2. 비교 결과

앞 절에서 설명한 다양한 왜곡에 대해서, 논문에서 제안하는 핑거프린트와 그것의 근간이 되었던 필립스 핑거프린팅 시스템의 핑거프린트 (PF로 표시) 와 성능을 비교하였다. 실험 결과는 그림 5.에 나타나 있으며 다양한 왜곡들에 대해서 논문에서 제안하는 핑거프린트가 더 나은 성능을 보이거나, 최소 비슷한 성능을 나타냄을 알 수 있다.

3. 기본 핑거프린트의 성능과 최종 핑거프린트의 성능

이 논문에서 사용한 기본 핑거프린트의 각각의 성능과 최종 핑거프린트의 성능을 살펴보면 아래 그림 6.과 같다. 그림 6.에서는 여러 가지 왜곡 중에서 BPF+MP3에 대한 성능을 나타내었다. 그림에서 보여지듯이, 5개의 기본 핑거프린트는 서로 다른 성능을 보이고 있으나, 큰 차이는 나지 않는다. 그리고 이 논문에서 제안하는 핑거프린트는 분명

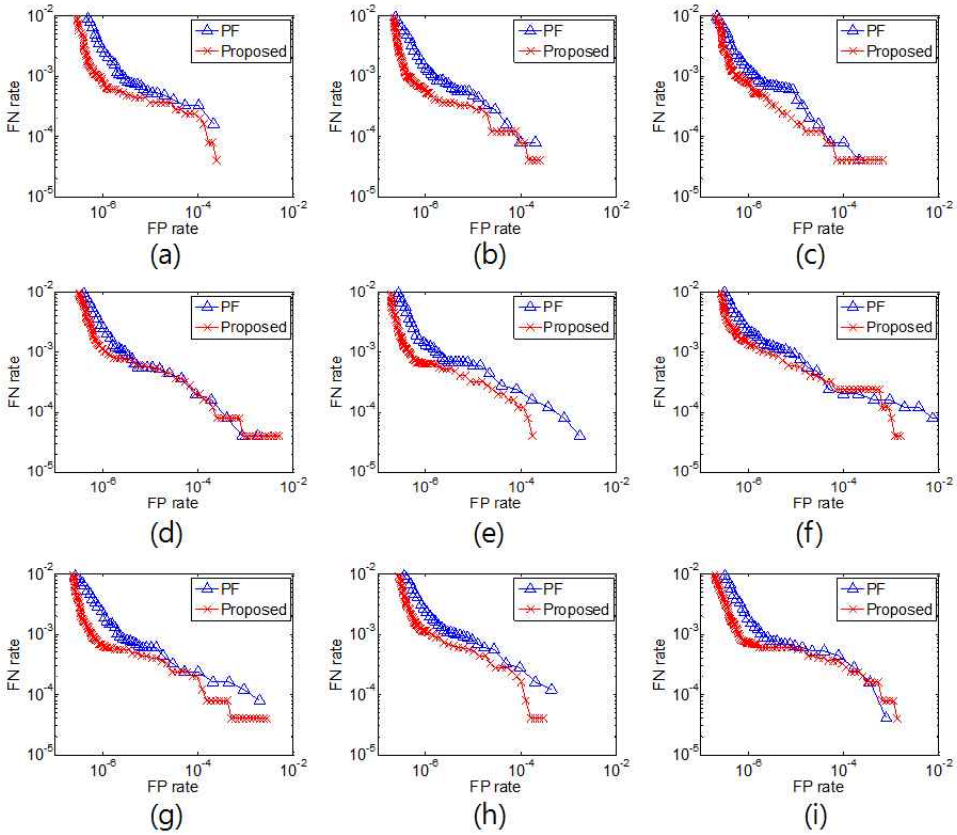

그림 5. 필립스 시스템의 핑거프린트와의 성능 비교: (a) TD+MP3, (b) EQ1+MP3, (c) V+MP3, (d) E+BP3, (e) BPF+MP3, (f) WMA+MP3, (g) EQ2+MP3, (h) EQ2+WMA+MP3, (i) EQ1+V+BPF+WMA+MP3

Fig. 5. Comparative results with Philips fingerprint: (a) TD+MP3, (b) EQ1+MP3, (c) V+MP3, (d) E+BP3, (e) BPF+MP3, (f) WMA+MP3, (g) EQ2+MP3, (h) EQ2+WMA+MP3, and (i) EQ1+V+BPF+WMA+MP3

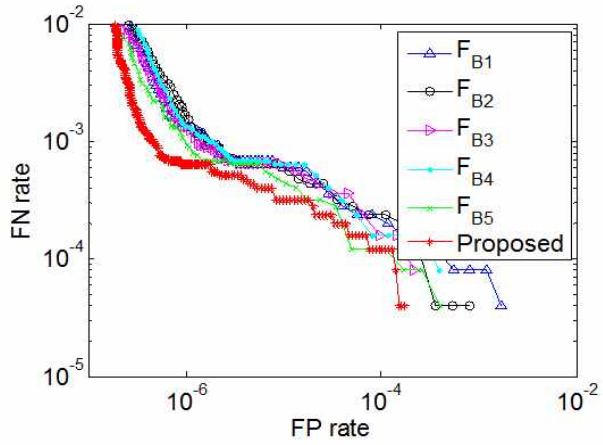

그림 6. 기본 핑거프린트들과의 성능 비교

Fig. 6. Comparative result with base fingerprints

5개의 이진 핑거프린트에 기초해서 만들어졌지만, 그것들의 성능을 뛰어넘는다는 사실을 알 수 있다.

4. 거리 계산 방법에 따른 성능 차이

제안한 핑거프린트를 활용하기 위한 거리 계산 방법이 IV장에 설명되어 있지만, 제안된 핑거프린트는 이진 핑거프린트의 일반적인 특성을 가지고 있으며, 이진 핑거프린트의 거리 계산에 많이 사용되는 해밍 거리 (Hamming distance) 를 사용하여 거리를 계산할 수 있다. 이를 위해서 쿼리 오디오로부터 $\vec{S}$를 계산한 후 이진화하는 과정 (III-3절)을 거쳐야 할 것이다. 그런 과정을 거친 후 해밍 거리를 사

용해서 얻은 결과의 성능과 제안한 거리 계산 방법으로 얻은 결과를 성능을 그림 7.에서 비교한다. 그림 7.에서는 그림 6.과 마찬가지로 여러 가지 왜곡 중에서 BPF+MP3에 대한 성능을 나타내었다. 그림에서 볼 수 있듯이 거리 계산 방법에 따라서 서로 다른 성능을 보임을 알 수 있다. 해밍 거리를 사용했을 경우에는 제안하는 방법보다 성능이 떨어지며, 다섯 개의 기본 핑거프린트를 결합하는 복잡한 방법을 사용했음에도 불구하고 기본 핑거프린트 중 하나인 필립스 핑거프린트보다 더 좋은 성능을 내지 못 한다. 다수개의 이진 핑거프린트를 다수투표 방법으로 결합했다고 하더라고, 거리를 어떻게 구하느냐에 따라서 성능의 차이가 많이 나며, 논문에서 제안하는 거리 계산 방법을 사용하는 것이 성능 향상에 많이 영향이 있음을 알 수 있다.

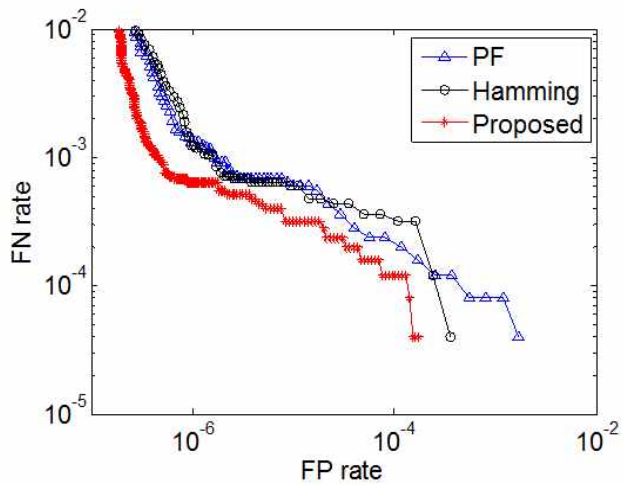

그림 7. 해밍 거리와의 성능 비교
Fig. 7. Comparative result with the Hamming distance

VI. 분석 및 보완점

제안한 핑거프린트는 여러 이진 핑거프린트를 합쳐서 만들었다는 점에서 특이성을 가진다. 분류기에서는 부스팅 알고리즘 (boosting algorithm), 전문가망 (mixture of experts)와 같이 결합을 통해서 성능을 향상시키려는 시도가 많이 있어왔다[14,15]. 또한 핑거프린팅 분야에서도 추출 과정에서 여러 필터들을 결합하여 성능을 향상시키려는 시도가 있었다[7,8]. 하지만 이 논문에서처럼 이진화된 핑거프린트 자체를 결합하려는 시도는 없었고, 이 논문에서는 그러한 시도를 통해서 핑거프린팅 시스템의 성능을 향상시킬 수 있다는 새로운 가능성을 제시하였다.

논문에서 제안한 방법은 핑거프린팅 시스템의 식별 성능을 개선시킬 수는 있지만, 기존의 이진 핑거프린트를 사용하는 것에 비해서 계산시간이 더 많이 필요하다는 단점이 존재한다. 핑거프린팅 시스템이 입력된 오디오에 대해서 핑거프린트를 추출하고, 그것을 정합하는 과정을 거치게 되고, 그것에 소요되는 시간을 각각 t_E, t_M이라고 하자. 전체 계산 시간은 $t_E + t_M$이 되는데, 일반적으로 $t_M \gg t_E$라 할 수 있다. 논문에서 제안한 핑거프린트를 추출하는 과정의 시간을 살펴보면, 핑거프린트를 뽑기 위한 정규화 과정과 변환 과정은 기존의 핑거프린트와 같지만 그 후 과정에 대해서 약 L배의 계산이 더 필요하다. 그림 2.에 나타난 것처럼 추출한 핑거프린트를 정합하기 위해서는 데이터베이스 검색 과정과 거리 계산 과정을 거치게 되는데 이 때 걸리는 시간은 $t_M = t_{DB} + N_{DB} \times t_d$라 할 수 있다. 이 때 N_{DB}는 데이터베이스 검색 과정에서 걸러진 부분집합의 원소갯수이고, t_{DB}는 데이터베이스 검색에 걸리는 시간, t_d는 각각의 거리 계산 과정에 걸리는 시간이다. 우리는 본 논문에서 핑거프린트 추출, 거리 계산에 대한 과정만 제안하였기에, 데이터베이스 검색에 대한 성능을 같다고 보고 기존의 필립스 핑거프린트에 대한 거리 계산 과정과 논문에서 제안하는 핑거프린트의 계산 과정을 비교한다. 기존의 필립스 핑거프린트의 거리 계산 과정은 각 비트별로 XOR 연산을 하는 것이고, 제안하는 핑거프린트의 거리 계산은 제곱 계산이다. 하지만, 제곱 계산에 대해서는 $\vec{S}$가 $\{0,1,2,3,4,5\}$대신 5bit의 신호를 가진다고 가정할 수 있고, 이를 이용하면 $\{0,1\}^6 \rightarrow \{0,1,4,9,16,25\}$의 연산으로 대체할 수 있다. 구현방법에 따라 차이는 있겠지만, t_d는 필립스 핑거프린트를 사용하는 것에 비해서 수 배 커진다고 할 수 있다. 즉 핑거프린트 추출과 거리 계산, 두 경우 모두 많은 계산을 필요로 한다. 이처럼 논문에서 제안한 핑거프린트는 시간적 성능에 대한 단점이 존재하기에, 계산 능력이 충분한 경우에 성능 향상을 위해 사용하는 것이 좋

을 것이다.

결합되는 기본 핑거프린트들이 하나의 콘텐츠에 대해서 서로 어느 정도 다른 비트를 추출해주어야 좋은 성능이 보일 것으로 추정된다. 마치 부스팅 알고리즘에서와 같이 [14], 결합이 되는 기본 핑거프린트들이 다른 기본 핑거프린트가 좋은 성능을 보이지 못하는 부분에서 좋은 성능을 보여야 결합에서 효과가 날 것이다. 어떤 쿼리 오디오 A,B를 핑거프린팅 시스템에서 식별하려 할 때, 각각 $\overrightarrow{F_{B1}}$, $\overrightarrow{F_{B2}}$, $\overrightarrow{F_{B3}}$, $\overrightarrow{F_{B4}}$, $\overrightarrow{F_{B5}}$를 핑거프린트로 사용하여 오디오 A,B에 대해서 공히, 5개 중 3개는 식별이 성공하고 2개는 실패하였다면, 이들을 결합하여 식별을 성공할 가능성이 높다. 하성공할 오디오 A는 5개 다 식별 성공할 오디오 B는 5개 다 식별 실패, 이런 결과라면 결합을 하더라도 오디오 B는 식별이 불가능할 것이다. 이렇듯 각각 식별능력을 가지고 있는 기본 핑거프린트들이 어느 서로 너무 유사하지 않도록 설정하여야 좋은 성능을 보일 것이다. 표 3.에서는 기본 핑거프린트들 간의 유사도를 비트 에러율로 나타내었다. 원본 오디오에 대해서 각각 $\overrightarrow{F_{B1}}$, $\overrightarrow{F_{B2}}$, $\overrightarrow{F_{B3}}$, $\overrightarrow{F_{B4}}$, $\overrightarrow{F_{B5}}$를 추출하고 서로간의 해밍 거리를 전체 길이로 나누었다. 이중 $\overrightarrow{F_{B1}}$과 $\overrightarrow{F_{B4}}$, $\overrightarrow{F_{B2}}$와 $\overrightarrow{F_{B4}}$가 비트 에러율이 0.06, 0.08로 상당히 유사하다는 점이 확인되고 있긴 하지만, 그 외에는 전체적으로 14% 이상의 비트가 서로 다른 값을 가지고 있었다.

표 3. 기본 핑거프린트들 간의 비트 에러율 (서로 간의 해밍 거리/전체 길이)
Table 3. Bit error rate between base fingerprints (Hamming distance/length)

	$\overrightarrow{F_{B2}}$	$\overrightarrow{F_{B3}}$	$\overrightarrow{F_{B4}}$	$\overrightarrow{F_{B5}}$
$\overrightarrow{F_{B1}}$	0.14	0.28	0.06	0.38
$\overrightarrow{F_{B2}}$	.	0.14	0.08	0.25
$\overrightarrow{F_{B3}}$	.	.	0.22	0.17
$\overrightarrow{F_{B4}}$	.	.	.	0.33

본 논문에서 제안한 방법에서 보완해야 할 몇 가지 내용들이 있다. 본 논문에서는 기존의 이진 핑거프린트와 비슷한 특성을 가지는 기본 이진 핑거프린트들을 만들었지만, 그것들을 어떻게 구성할 지에 대해서는 추가적인 연구가 필요하다. 즉, 이 논문에서는 결합을 통해서 성능이 올라갈 수 있다는 가능성을 보여주었지만, 항상 그렇게 성능을 향상시킬 수 있기 위해서 어떤 과정이 필요한지에 대한 연구가 필요하다. 부스팅 알고리즘에서처럼 순차적으로 좋은 성능을 가지는 핑거프린트를 학습하여 결정할 수 있다면 좋을 것이다. 또한 거리를 계산하기 위해서 제곱값을 이용하였지만, 제곱 이외의 다른 방법을 찾는 것도 차후의 연구로 가치가 있다. 합벡터의 값이 왜곡에 대해서 변하는 정도에 대한 통계치를 구해서 그것을 거리 계산에 활용하면 더욱 나은 성능을 얻을 수 있을 것으로 기대된다. 현재 5가지 기본 핑거프린트를 사용하였는데, 기본 핑거프린트의 개수를 늘려가면서 성능을 살펴보는 것 또한 논문에서 제안하는 방법의 성능을 검증하는데 주요한 지표가 될 것이다.

VII. 결론

논문에서는 오디오 핑거프린트를 추출하는 새로운 방법을 제안하였다. 기존의 방법들과는 달리 다수 개의 기본 이진 핑거프린트를 결합하여 하나의 이진 핑거프린트를 만드는 방식을 사용하였다. 거리 계산을 위해서 기존에 이진 핑거프린트에 많이 사용되는 해밍 거리를 사용하지 않고, 기본 이진 핑거프린트들의 합의 제곱을 이용하는 방법을 사용하였다. 실험 결과, 기본 이진 핑거프린트들에 비해서 향상된 성능을 보임을 알 수 있었다. 여러 개의 이진 핑거프린트를 결합하여 각각의 성능보다 더 나은 성능을 낼 수 있는 가능성을 제시하였으며, 논문에서 제안한 방법을 이용해서 새로운 이진 핑거프린트를 만들거나 기존의 이진 핑거프린트의 성능을 강화할 수도 있을 것이다.

참 고 문 헌

[1] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," Journal of VLSI signal processing, Vol. 41, No. 3, pp. 271-284, 2005

[2] J. Haitsma and T. Kalker, “A highly robust audio fingerprinting system,” in Proc. Int. Conf. Music Information Retrieval, Paris, France, 2002, pp. 107 – 115.
[3] C. Burges, J. Plat, and S. Jana, “Distortion discriminant analysis for audio fingerprinting,” IEEE Trans. Speech Audio Process., vol. 11, no. 3, pp. 165 – 174, May 2003.
[4] J. S. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. D. Yoo, “Audio fingerprinting based on normalized spectral subband moments,” IEEE Signal Process. Lett., vol. 13, no. 4, pp. 209 – 212, Apr. 2006.
[5] Y.Ke, D. Hoiem, and R. Sukthankar, “Computer vision for music identification,”in Proc. CVPR, 2005, vol. 1, pp. 597 – 604.
[6] K. Covell and S. Baluja, “Known-audio detection using waveprint: Spectrogram fingerprinting by wavelet hashing,” in Proc. ICASSP, Hawaii, pp. 237 – 240., 2007
[7] S. Kim and C. D.Yoo, “Boosted binary audio fingerprint based on spectral subband moments,” in Proc. ICASSP, Hawaii, pp. 241 – 244, 2007
[8] D. Jang, C. D.Yoo, S. Lee, S. Kim, and T. Kalker, "Pairwise boosted audio fingerprint," IEEE Trans. on Information Forensics and Security, Vol. 4, Iss. 4, pp 995-1004, 2009
[9] M.S. Park, H.R. Kim, and S.H. Yang, “Frequency-temporal filtering for a robust audio fingerprinting scheme in real-noise environments,” ETRI Journal, vol. 28, no. 4, pp. 509 – 512, Aug. 2006.
[10] Y. Liu, K. Cho, H. S. Yun, J. W. Shin, and N. S. Kim, "DCT based multiple hashing technique for robust audio fingerprinting," in Proc. ICASSP, 2009
[11] D. Jang, C. D. Yoo and T. Kalker, “Distance metric learning for content identification,” IEEE Trans. on Information Forensics and Security, Vol. 5, Iss. 4, pp. 932-944, 2010.
[12] M. Jin and C. D. Yoo, "Quantum hashing for multimedia," IEEE Trans. on Information Forensics and Security, Vol. 4, Iss. 4, pp. 982-994, 2009
[13] P. J. O. Doets, "Distortion estimation in compressed music using only audio fingerprint," IEEE Trans. on Audio, Speech, and Language Processing, Vol. 16, Iss. 2, pp. 302-317, 2008
[14] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," Machine Learning, vol. 37, no. 3, pp. 297 – 336, 1999.
[15] C. M. Bishop, Pattern recognition and machine learning, Springer, 2006

저 자 소 개

장 달 원

- 2010년 2월 : KAIST 전기 및 전자공학과 박사
- 2010년 11월 ~ 현재 : KETI 디지털미디어연구센터 선임연구원
- 주관심분야 : 음악정보검색, 오디오인식, 기계학습

이 석 필

- 1990년 2월 : 연세대학교 전기공학과 학사
- 1997년 8월 : 연세대학교 전기공학과 박사
- 1997년 2월 ~ 2002년 2월 : 대우전자(주) 선임연구원
- 2002년 3월 ~ 현재 : KETI 디지털미디어연구센터 센터장
- 주관심분야 : 방송통신융합기술, 실감오디오기술, 음악정보검색, AI 등