楽曲検索システムにおける音響フィルタの影響

高田　怜† 三浦　亮† 喜田　拓也†

† 北海道大学大学院情報科学研究科　〒 060–0814 北海道札幌市北区北１４条西９丁目
E-mail: †{takada,ryo_miura,kida}@ist.hokudai.ac.jp

あらまし　歌声合成システムの普及に伴い，消費者であるユーザ自身が作成した楽曲がインターネット上に爆発的に多く流通するようになった．膨大な楽曲データベースから，クエリとする音響信号に類似する部分を高速に検索することは，高度な楽曲推薦システムの実現などにつながる．2011 年に Xiao らは，大規模な楽曲データベースに対して，音楽指紋上のハミング距離に基づく高速な類似楽曲部分検索手法を提案し，一つの検索システムを実現している．本稿では，Xiao らの楽曲検索システムにおいて，歌声合成システムを用いて作成された VOCALOID 楽曲と呼ばれるデータに対する検索性能を評価するとともに，歌声の音域をフィルタリングした楽曲信号を用いることで，検索システムの性能にどのような影響があるかについて論じる．

キーワード　楽曲検索システム，音楽指紋，ボーカロイド曲，データ・リダクション

Influence of Acoustic Filter upon Music Retrieve System

Satoru TAKADA†, Ryo MIURA†, and Takuya KIDA†

† Graduate School of Information Science and Technology, Hokkaido University
Kita 14jo, Nishi 9, Sapporo, 060–0814 Japan
E-mail: †{takada,ryo_miura,kida}@ist.hokudai.ac.jp

Key words　music retrieval, audio fingerprints, VOCALOID song, data reduction

1. はじめに

近年のインターネットの発展に伴い，膨大なマルチメディア情報を個人で取り扱うことは特別なことではなくなった．中でも音楽データに関しては，iTunes Store [1] などのインターネット販売サイトが一般的に普及し，楽曲をインターネット経由で入手する人が増加している．

また一方で，YouTube [9] やニコニコ動画 [13] といった，消費者生成メディア（Consumer Generated Media: CGM）サイトの成長も著しい．これまでは企業が提供する楽曲を購入するのみであったユーザが，それら CGM サイト上に自身が作成した楽曲を公開するようになった．特に，クリプトン・フューチャー・メディア（株）から発売された初音ミク [19] に代表される VOCALOID と呼ばれる一連の音声合成ソフトウェアは，アマチュア作曲家による楽曲創作活動を大きく後押しすることになった [20]．そのようなユーザ生成コンテンツ（User Generated Contents: UGC）として新たに生み出される楽曲は，今日プロが作成する楽曲よりも爆発的に増加している．

しかしながら，そうした UGC は適切なメタデータが整備されることが稀であることから，大量の UGC 楽曲データを管理し，ユーザが自身の好みにあった楽曲を見つけ出すことは容易ではない．

大量のアイテムからユーザの嗜好に合うものを推薦する有効な手法としては，協調フィルタリング [2], [15] がある．協調フィルタリングでは，各アイテムについてのユーザ評価を元に，類似した嗜好を持つユーザ群を特定し，その結果に基づいてアイテムの推薦を行う．あるいは，アイテムの類似度に着目した推薦を行う変種 [14] も提案されている．しかしながら，日々大量に生成され，十分な量のユーザ評価がなされない UGC の楽曲を対象に協調フィルタリングを行うことは難しい．

これに対し，楽曲の内容や付随するメタ情報の類似度を用いたコンテンツベースによる楽曲推薦手法 [6], [11], [16], [21] や，協調フィルタリングとの併用によるハイブリッド手法 [10], [12], [18] では，ユーザによる未評価の楽曲に対しても推薦可能であるため，有効に機能することが期待できる．コンテンツベースによる手法では，何らかの指標による類似楽曲の検索が重要になる．すなわち，膨大な楽曲データベースから類似楽曲を効率よく検索するために，何らかの音楽指紋 [3] を用いた高速な近似検索が必要となる．

多量の高次元データに対する効率よい近似検索手法として，Locality Sensitive Hashing (LSH) に基づく手法が近年注目されている [4], [7], [8]．Xiao ら [17] は，LSH による近似検索手法

にヒントを得て，Haitsma ら [5] が提案した音楽指紋に対する高速な近似検索システムを 2011 年に提案した．彼らの楽曲検索システムでは，任意の長さの音楽信号をクエリとして用いることができ，クエリに近い楽曲の部分を高速に検索することができる．

本稿では，楽曲検索システムを UGC である VOCALOID 楽曲のデータベースに適用し，その検索性能を評価する．今回，2000 曲の VOCALOID 楽曲をデータベースとし，同じ楽曲を人間が歌唱したデータ 50 曲をクエリとして用意した．

Xiao ら [17] の楽曲検索システムにおける結果として，クエリを楽曲の特定の一部分とした時に検索精度が悪化するということを確認した．これは，ボーカル部分の違いが影響していると考えられる．そこで，中音域の影響を減じる前処理を施すことで，検索精度の改善を行えるのではないかという予想を立て，その検証実験を行った．以降では，Xiao ら [17] のシステムを，著者らの頭文字をとって *XSK* **システム**と呼ぶことにする．

2. XSK システム

本節では，XSK システムの概要について述べる．XSK システムでは，楽曲に対する音声指紋のビット列をデータベースに保存しており，クエリに対する音声指紋と一致する楽曲を答えとして返す．このとき，クエリとなる音響信号データとしては，楽曲の全体である必要はなく，楽曲の一部分のみでも検索が可能となっている点が優れている．XSK システムは

（1） 楽曲データから音楽指紋を生成

（2） 音楽指紋上での探索

の二段階を踏んで楽曲検索を行なっている．それぞれについて説明する．

2.1 音楽指紋の生成について

音楽指紋とは，楽曲の音響信号データから作られる文字列（ビット列）のことで，その楽曲の特徴を数キロビット程度の比較的小さいデータ量で表現したものである（図 1）．通常，音楽指紋から元の音響信号に復元することはできないが，楽曲の同一性を軽量に判定するために用いられている．音楽指紋を用いると，一曲分の楽曲データ信号をそのままデータベースにした場合に比べて一曲分のデータ量が格段に小さくなり，また音楽ファイルの形式に依存しない検索システムが構築できる．XSK システムでは，Haitsma ら [5] によって提案された音楽指紋が用いられている．彼らの音楽指紋は，ハミング距離によるビット列の差異がなるべく実際の楽曲の差異となるように設計されている．

以下ではまず，Haitsma ら [5] による音楽指紋の具体的な構築方法について述べる．

まず，楽曲の先頭から順に，ある短い長さの区間で区切りながら音響信号を取り出し周波数解析を行う．我々のシステムでは，連続する区間のずれ幅（シフト量）を区間幅と同じとし，XSK システムでは，区間幅の $\frac{1}{32}$ としている．区間幅と同じでないということは，取り出す部分が連続する区間同士で重なるということであり，たとえば区間幅の $\frac{1}{32}$ のときは，区間幅を 320 ミリ秒とすると，区間のシフト量は 10 ミリ秒となる．この場合，3 分の楽曲に対して，18,000 個の区間が取り出される．先頭から i 番目の区間を，i 番目の**フレーム**と呼ぶことにする．取り出した各フレームについて，300Hz から 2000Hz の音を周波数に応じて 33 個の重複しない領域に分割し，各領域の強度を測る．ここで，i 番目のフレームの下から j 番目の周波数帯域の強度を $E(i,j)$ と書くことにする．また，$E(i) = \langle E(i,1), E(i,2), \cdots, E(i,33)\rangle^T$ とする．楽曲の全体にわたって上の手順を行うと，33 次元のベクトル $E(i)$ が並んだデータが得られる．いま，楽曲全体から取り出されるフレームの総数を N とし，これを**楽曲の長さ**とする．すなわち，i,j はそれぞれ，$[1,N],[1,33]$ の範囲の整数を取る．

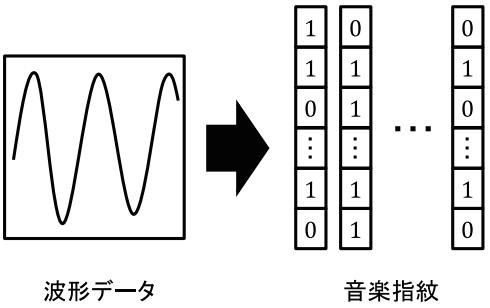

図 1　波形データから音楽指紋への変換

次に，$E(i)$ の隣接する成分同士の差分を求め，32 次元のベクトル $E'(i)$ を求める．すなわち，$E'(i) = \langle E(i,1) - E(i,2), E(i,2) - E(i,3), \cdots, E(i,32) - E(i,33)\rangle$ とする．さらに，$E'(i)$ から隣接するフレーム同士の差分を求め，$ED(i)$ とする．すなわち，$ED(i) = E'(i) - E'(i-1)$ である．ここで，便宜上，$E'(0)$ は零ベクトルとする．最終的に，$ED(i)$ の各成分の正負で二値化したベクトル $F(i)$ の並びを楽曲の音楽指紋とする．すなわち，

$$F(i,j) = \begin{cases} 1 & \text{if } ED(i,j) > 0, \\ 0 & \text{if } ED(i,j) \leqq 0. \end{cases} \tag{1}$$

ただし，ここで $F(i,j), ED(i,j)$ は，それぞれ $F(i), ED(i)$ の j 番目（$1 \leqq j \leqq 32$）の成分である．

各ベクトル $F(i)$ は，**サブ指紋**と呼ばれる．上述した構成方法から明らかなように，サブ指紋の一つ一つは 32 ビットのベクトルであり，単一のサブ指紋は楽曲の同定を行えるだけの情報を持っていない．そこで，検索する際は連続する複数のサブ指紋をまとめて取り扱う．このサブ指紋をまとめたものを**サブ指紋ブロック**と呼ぶ．XSK システムでは，128 個のサブ指紋で一つのサブ指紋ブロックを構成している．

2.2 音楽指紋の探索

次に，音楽指紋上での探索について概説する．

XSK システムでは，クエリとなる音響信号データから抽出したサブ指紋ブロックと，データベース中の最も類似したサブ指紋ブロックをもつ楽曲が検索結果として出力される．このとき，サブ指紋ブロック間の類似度にはビット誤り率が用いられる．二つのサブ指紋ブロック A, B のビット誤り率 BER は以下のように定義される．

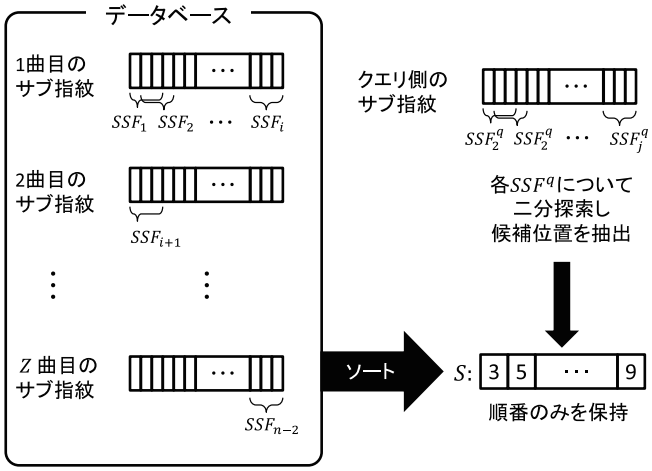

図 2　サブ指紋短系列 (SSF) による候補位置探索の流れ

$$BER(A,B) = \frac{\sum_{k=1}^{L} W_H(F_A(k) \oplus F_B(k))}{32L}$$

ここで，$\oplus$ は排他的論理和 (XOR) の論理演算子，$W_H(x)$ はベクトル x のハミング重み，L はサブ指紋ブロックを構成するサブ指紋の個数である．

XSK システムでは，まずクエリのサブ指紋を用いて楽曲データベース中の候補位置を絞り込んでから，サブ指紋ブロックに対するビット誤り率を計算する．しかし実際には，サブ指紋単体では楽曲の同定に十分な情報量を持っていないので，連続する 3 個のサブ指紋を並べた短いサブ指紋の系列（96 ビット）を考える．論文 [17] では，この短いサブ指紋の系列を**サブ指紋短系列** (Sequence of sub-fingerprint: SSF) と呼んでいる．すなわち，楽曲データベース中の全曲から得られたサブ指紋の系列を $FP = (FP_1, FP_2, \cdots, FP_n)$ とすると，この楽曲データベースは $n-2$ 個の SSF，$SSF_1 = (FP_1, FP_2, FP_3), SSF_2 = (FP_2, FP_3, FP_4), \cdots, SSF_{n-2} = (FP_{n-2}, FP_{n-1}, FP_n)$ が並んでいると考える．

SSF を探索する流れについてを図 2 に示している．まず，前処理として楽曲データベース中に含まれる SSF をソートしておく．ただし，SSF を列挙して保持していては 3 倍の領域が必要となるので，実際には各 SSF のソート位置を表す 1 次元配列 $S = S_1, S_2, \cdots, S_n$ を計算する．つまり，配列 S は，FP に対する長さ 3 で制約を付けた接尾辞配列と見ることができる．

与えられたクエリに対し，FP と同様に SSF の列を計算する．そして，クエリ側の全ての SSF について配列 S 上で二分探索を行う．また，二分探索された位置の前後を調べることで，SSF の近傍検索をすることができる．以上の手続きで求められた候補位置を開始点とするサブ指紋ブロックを考え，クエリとのビットエラー率を計算し，その結果をソートして上位のものを検索結果として出力する．

2.3　XSK システムの検索精度

ここでは，XSK システムの検索精度の調査実験を行う．

実験に用いるデータとして VOCALOID に関する楽曲を用いる．データベースとして 2283 曲の VOCALOID 楽曲，クエリとして同じ楽曲を人間が歌唱したデータ 50 曲を用いる．実験に用いるデータとして VOCALOID の楽曲を用いた理由は，同じ曲でボーカル違いの曲を多く集めるのに VOCALOID オリジナル曲が適していたということ，また，近年音声合成の研究が非常に盛んに行われていることなどが挙げられる．また，クエリの 50 曲は，楽曲全体と，楽曲のサビの部分を抽出したものの 2 つを用いる．サビ部分の抽出は人力で行った．

表 1　XSK システムの検索精度 (50 曲).

	クエリ	
	全体	サビのみ
正答率	90%	56%

表 2　XSK システムの検索精度 (28 曲).

	クエリ		
	全体	サビのみ	ボーカル無し
正答率	89%	46%	82%

以上のデータセットによる XSK システムの検索精度は表 1 のようになった．

楽曲全体をクエリとした時の検索の正答率は，90%と精度の良い結果となった．しかし，検索が上手く行われなかった楽曲もある．検索に失敗したのは，クエリの楽曲が元の楽曲の調を変えて (歌いやすいように楽曲全体の音の高さを変えて) 歌われたためである．音楽指紋は楽曲の音の絶対的な高さから作られているため，元の楽曲と比べて全体の音の高さを変えてしまうと作られる音楽指紋が大きく変わってしまい，検索が上手く行われなくなってしまう．また，クエリをサビのみとした時に，全体の時と比べ検索精度の低下がみられた．その原因としては，正解の楽曲とクエリの楽曲の大きな違いであるボーカルである．楽曲全体をクエリとした時は，前奏や間奏など，ボーカルの無い部分は元の楽曲と全く同じである．このため，その部分で検索が上手く行われているものと考えられる．しかし，サビ部分のみを取り出した時は，クエリ全体がボーカルのある部分のため，検索にヒットするべき元の楽曲と全く同じ部分が存在しない．そのため，検索精度が劣ってしまったと考えることが出来る．

そこで，前奏や間奏などの，ボーカルの無い部分を切り出して同様の実験を行った．この時，ボーカルの無い部分で検索に十分な量を取り出せない楽曲があったため，取り出すことのできた 28 曲のみで精度調査を行った．そのときの結果が表 2 である．ボーカルの無い部分の時の正答率はサビのみの時よりも良い結果となったため，ボーカルが検索精度を低下させている原因であると確認できた．全体の時と比べ精度が下がっているのは，クエリ自体が短く，検索に使用できる情報が少ないことが原因であると考えらえれる．

3. 提 案 手 法

クエリをサビのみとした時に，検索精度の改善をしたい．先に述べたように，検索精度の悪化の原因がボーカル部分であるなら，ボーカルのある帯域の音の高さの強さを弱めれば良いのではないかという予想ができる．また，楽曲全体でも検索でき

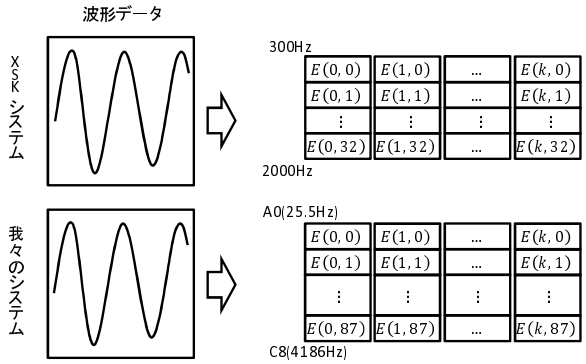

図 3　解析方法の違い

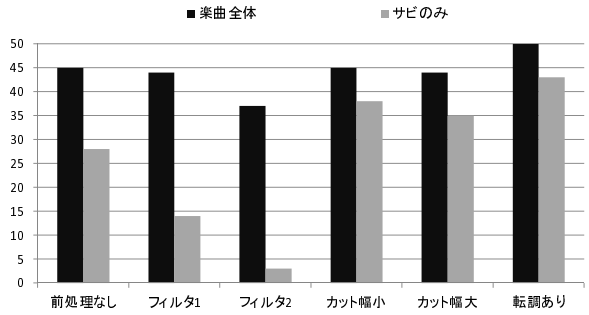

図 4　各検索実験における正答数

なかった楽曲については，検索できなかった原因が楽曲全体の移調によるものと考えられる．よって，これらの問題に対応するような音楽指紋を構築する必要がある．

XSK システムの音楽指紋生成部分には直接手を加えることが出来なかったため，今回我々は，XSK システムと同じ尺度で検索が行えるプログラムを構築した．以下では，実装したプログラムで用いる音楽指紋の構成方法について述べる．

3.1　構築したプログラムについて

大筋は XSK システムで取り入れられている Haitsma ら [5] の生成法と同様であるが，以下のような点で異なった生成法をとる．

区間のずらし幅について，楽曲の短い区間を取り出すということは変わらないが，我々のプログラムでは，区間のずらし幅は区間長と同じとしており，取り出す部分をオーバーラップさせない．そのため，XSK システムと区間長が同じであったとしても，生成される音楽指紋の数は少なくなる．

取り出す周波数帯について，取り出す周波数帯の高さをピアノの鍵盤に対応するように音階ごとに区切るようにする．区切られる領域数は，XSK システムでは 33 であったのに対し，ここでは 88 とする．この 88 に分割されたの領域から適当な部分を取り出し，ベクトルの列を形成する．このベクトルの列が XSK システムでの行列 E に相当する．(図 3 を参照) これにより，後述する転調を考慮した検索が可能となる．

次に，我々のプログラムにおける音楽指紋の探索について述べる．XSK システムの探索法と同じく，SSF を用いて探索を行う．クエリの音楽指紋の全ての SSF について，データベースの音楽指紋の SSF を線形に探索する．クエリの各 SSF と，最もビット誤り率の小さい SSF を持つデータベース中の楽曲についてランキングをとり，最もヒット回数が多かった楽曲を検索結果として出力する．我々のプログラムでは，サブ指紋ブロックは使用しない．

3.2　検索精度の調査実験

独自のプログラムを実装したので，先に立てた予想がどの程度正しいものであったかの調査実験を行う．

本稿で行う評価実験では，複数の環境で音楽指紋を生成し，検索精度を調査する．それぞれについて述べる．

- フィルタ

データベース，クエリの全ての楽曲に特定の帯域の音の強さを制限するようなフィルタを通した後に，音楽指紋を生成する．フィルタは MATLAB のライブラリにある fir1 関数と filter 関数を用いる．fir1 関数により特定の帯域の音の強さを抑えるようなフィルタを設計する．その後，filter 関数により元の音源にフィルタリングを行う．これにより，楽曲の一定の高さの音が小さくなったり，聞こえなくなったりする．ここでは，フィルタリングする帯域として 500Hz から 1000Hz(フィルタ 1)，100Hz から 3000Hz(フィルタ 2) という二つの設定をとる．この音楽指紋による実験は XSK システムを用いる．

- カット

ボーカルがあると考えられる中音域の高さの音をはじめから使用せず，データベース，クエリの全ての楽曲について，高音域と低音域の音のみを用いて音楽指紋を生成する．ここでは，D4#(311Hz) から D5(587Hz)(カット幅小)，C2#(69Hz) から G6(1568Hz)(カット幅大) の高さの音を使用しないという二つの設定をとる．この音楽指紋による実験は我々のプログラムを用いる．

- 転調あり

我々のプログラムでは，前述したように楽曲を解析する際には，ピアノの鍵盤に対応するように音を区切って音楽指紋の生成に利用している．そこで，データベースはそのまま，クエリのみ，楽曲全体で音楽指紋の生成に使用する部分を 1 つずつずらしながら，クエリとなる音楽指紋を生成する．これにより，楽曲全体の高さが変えられている楽曲の検索についても対応できる．

3.3　提案手法の評価

上記の環境において生成した音楽指紋上で，検索精度の調査実験を行った．結果を図 4 にまとめた．

前処理を行わなかった時との比較をすると，クエリを楽曲全体としたときは大きく精度は変わらなかった．しかし，クエリをサビのみとしたときに精度に大きき変化がみられた．フィルタについては，楽曲全体，サビのみのどちらでも検索精度は下がった．特に，フィルタの幅が大きい時により精度が下がる結果となった．ここで使用されている音楽指紋は，音の上下と前後の強さの差により生成されている．そのため，フィルタをかけて音の強さを平坦にしてしまったことにより，作られる音楽指紋が大きく変化してしまったためと考えられる．また，カットについては，前処理をしない時と比べ精度が上がるという結果となった．これは，やはりサビのみでの検索精度の悪化は中音域にあり，その部分を検索に使用しないことで，検索精度を

改善できるということが確認できた．転調を考慮した検索については，楽曲全体ではクエリの 50 曲全てで検索に成功した．これにより，調が変えられたことによる検索の不成功は，音楽指紋を作り出す楽曲のデータをずらせばよいということが確認できた．また，同様にサビのみでの検索も精度が改善した．

4. おわりに

本稿では，楽曲検索システムにおける特定音域をフィルタリングした際の検索精度の変化を調査した．今回，特定音域をフィルタリングすることは検索システムに悪影響を及ぼすことが判明した．しかし，フィルタリングするのではなくはじめから特定の情報を検索にあえて使用しないことで，検索精度の改善を行えることが確認できた．また，転調に対応する処理を加えることで，既存の検索手法よりも精度の良い検索が行えることを確認した．

今後は，100 万曲を超える巨大なデータセットでの検索性能について調査を行い，より高速・高精度な検索システムの実現を模索することが課題である．

5. 謝　　辞

本研究を行うにあたり，プログラムの提供をして頂いた北研二氏 (徳島大学) に感謝する．

文　　献

[1] Apple.com. itunes store. http://www.apple.com/jp/itunes.

[2] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI'98*, pages 43–52, 1998.

[3] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of audio fingerprinting. *J. VLSI Signal Process. Syst.*, 41(3):271–284, November 2005.

[4] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[5] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *ISMIR 2002*, 2002.

[6] Keiichiro Hoashi, Kazunori Matsumoto, and Naomi Inoue. Personalization of user profiles for content-based music retrieval based on relevance feedback. In *Proceedings of the eleventh ACM international conference on Multimedia*, MULTIMEDIA '03, pages 110–119, New York, NY, USA, 2003. ACM.

[7] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2130 –2137, 29 2009-oct. 2 2009.

[8] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. NIPS, 2009*, pages 1042–1050, 2009.

[9] YouTube LLC. Youtube. www.youtube.com/.

[10] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning Similarity from Collaborative Filters. In *International Society of Music Information Retrieval Conference*, pages 345–350, 2010.

[11] Brian McFee and Gert R. G. Lanckriet. Heterogeneous embedding for subjective artist similarity. In *ISMIR'09*, pages 513–518, 2009.

[12] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[13] niwango. ニコニコ動画. www.niconico.jp/.

[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Itembased collaborative filtering recommendation algorithms. In *Proc. 10th International Conference on the World Wide Web*, pages 285–295, 2001.

[15] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[16] Malcolm Slaney, Kilian Q. Weinberger, and William White. Learning a metric for music similarity. In *ISMIR'08*, pages 313–318, 2008.

[17] Qingmei Xiao, Motoyuki Suzuki, and Kenji Kita. Fast hamming space search for audio fingerprinting systems. In *ISMIR*, pages 133–138, 2011.

[18] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):435 –447, feb. 2008.

[19] 博之 伊藤. 初音ミク as an interface (特集 CGM の現在と未来 : 初音ミク, ニコニコ動画, ピアプロの切り拓いた世界). **情報処理**, 53(5):477–482, may 2012.

[20] 真孝 後藤. 初音ミク, ニコニコ動画, ピアプロが切り拓いた CGM 現象 (特集 CGM の現在と未来 : 初音ミク, ニコニコ動画, ピアプロの切り拓いた世界). **情報処理**, 53(5):466–471, may 2012.

[21] 藤原 弘将 and 後藤 真孝. Vocalfinder : 声質の類似度に基づく楽曲検索システム (検索・推薦). **情報処理学会研究報告**. *[**音楽情報科学**]*, 2007(81):27–32, 2007-08-01.