E-007

スペクトルディップとクロマベクトルの併用による和音推定
Chord Estimation by Combination of Chroma Vector and Spectrum Dip

黒川 奈桜子† 斎藤博昭†

概要 和音推定の既存手法であるクロマベクトル法とスペクトルディップ法の 2 手法を組み合わせることで，2 手法の弱点を互いに補い合い，より高精度な和音推定のシステムの開発を行う．実験では 96 パターンの単和音とピアノ楽曲を対象に，各フレームで正解の和音を出力する平均順位を求めた．単和音ではクロマベクトル法 3.80 位，スペクトルディップ法 4.19 位に比べ提案手法は 2.98 位となり，楽曲でもクロマベクトル法 15.18 位，スペクトルディップ法 11.57 位に対して提案手法は 11.31 位となった．

1. はじめに

和音推定を扱う研究では音楽音響信号を用いたものがほとんどだが，これらは大きく「和音の構成音それぞれの音高を求め和音を推定する」ものと「スペクトルから直接和音を推定する」[1]ものに二分される．本研究は後者に属するものだが，スペクトルから直接和音を推定する研究の多くは『クロマベクトル』[2]と呼ばれる特徴量を用いている．クロマベクトル法は，同音名のパワーを 12 音名分それぞれに加算してパワーの強い音程を組み合わせてコードを推定するというところから直観的でノイズに強くはあるが倍音成分が大きい音響信号に対しては誤りが多くなるという弱点を持つ．改善手法として和音の遷移確率の学習[3]-[5]やクロマベクトルの次元圧縮を行った研究[6]もあるが，根本的にはクロマベクトルの精度に依存している．

クロマベクトルの弱点を鑑みて，それに代わる新たな特徴量『スペクトルディップ』を定義しそれによって推定精度の向上を目指した研究もある[7][8]．クロマベクトル法が周波数の言わば山を見ているのに対し，スペクトルディップ法は周波数の谷すなわちパワーの小さい周波数によって和音を特徴づけしている．こうすることで倍音成分が大きくなっても精度に影響が出づらく高速フーリエ変換（FFT）のポイント数を減らしても精度が落ちにくいが，谷を埋めてしまうようなノイズを含む入力においては精度が低くなってしまう．

以上のことを踏まえ本研究では，クロマベクトル法とスペクトルディップ法を組み合わせることで互いの不得手な部分を補い，和音推定の更なる精度向上を目的とする．

2. 和音推定

先に述べた二手法，クロマベクトル法とスペクトルディップ法について詳しく述べ，各手法の特徴について考える．

2.1 クロマベクトル法

どちらの手法でも音響信号の周波数分布を用いるため，入力された音響信号にはまず FFT を行う．FFT によって得られたスペクトルから，クロマベクトル法では 12 音それぞれに対して同音名のパワーをオクターブは問わずに合算する。こうして作られた 12 次元のベクトルを「クロマベクトル」と呼ぶ．12 音の中からパワーの大きいものをいくつか選んで和音を作るなどして和音推定を行っている．この方法なら多少のノイズがあっても問題ないが，図 2.1 のように単音のみの入力を与えた場合でも倍音成分が含まれるために複数の音が鳴っているように分析されてしまう．またもう一つの問題点として，FFT の解像度が高くなければならないということが挙げられる．スペクトルの解像度が低いと，低周波域の解像度が下がってしまう．すると，低周波域に発音があった場合，C の音が鳴っているにも関わらず，隣の C#にまでパワーが足されてしまうというようなことが発生する．

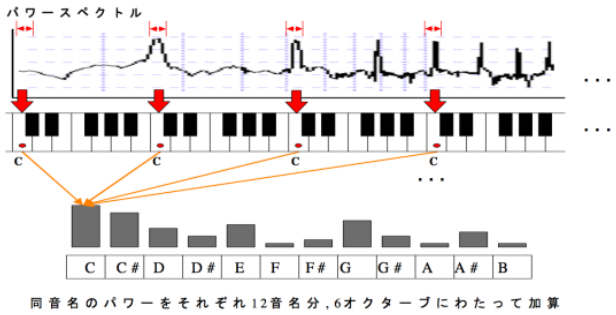

図 2.1 C 単音をクロマベクトルで表した場合

2.2 スペクトルディップ法

スペクトルディップ法は，異なる和音であれば谷となる周波数（スペクトルディップ）も異なることを利用している．先に和音ごとに谷となる周波数を見つけてリスト化しておき，入力された音響信号のスペクトルをそのリストと照らしあわせて和音の推定を行うのである．つまりスペクトルディップ法のスコアは値が小さいほど，その和音「らしい」ということになる．スペクトルディップは和音の構成音のオクターブによらず固定であり倍音の影響を受けにくい。クロマベクトル法で行っている『音名ごとの合算』という作業を行わないため実行も速い．しかしスペクトルの谷，つまりは音が鳴っていないということを特徴量としているため和音を構成していない音を含む入力には弱い．旋律や打楽器などといった音によって谷が埋まってしまうと推定の精度が著しく低くなってしまうのである．

3. 提案手法

以上のことを踏まえ，クロマベクトル法とスペクトルディップ法それぞれのスコアを算出・統合し和音推定の精度を上げることを考えた．

†慶應義塾大学大学院理工学研究科 Graduate School of Science and Technology, Keio University

3.1 概要

全体の流れ図を図 3.1 に示す．入力は wav ファイルとし，可能性が高いとされた順に和音を出力する．本実験では予め用意しておいた音源を用いたが将来的にはリアルタイムでの実行を目指しているため，なるべく動作が軽いスペクトルディップ法を主に使うように設計している．関連研究としてクロマベクトルと学習を用いたものもあるが[2]-[4]，本研究では機械学習は用いない．また，楽曲に対する実験も行うがコード進行は考慮に入れず，単位時間ごとに独立した推定を行う．これは、あくまでメロディーなど和音を乱す音の影響を調べるために実験を行うからである。

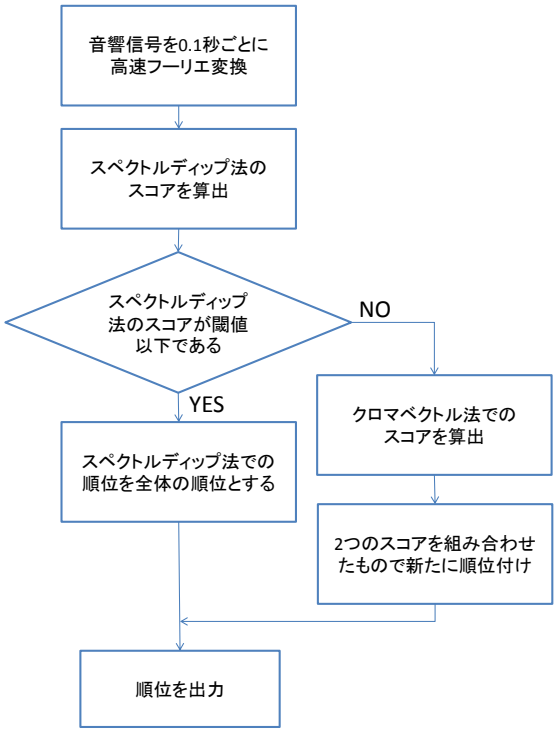

図 3.1　処理の流れ

以下に実験に用いた各手法の仕様及びスコアの統合式を述べる．

3.2 スペクトルディップ法

本研究では蔵内が作成したディップリスト[9]を使用している．各和音におけるスペクトルディップのリストは図 3.2（縦軸：パワー，横軸：周波数）のようなモデルによって 0Hz～4186Hz の範囲で求められている．表 3.1 はその一例である．

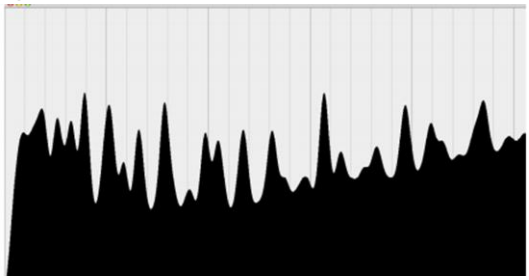

図 3.2　ディップリスト作成に用いられた
Cmaj の指数関数モデルスペクトル

表 3.1　ディップリストの例

和音名	スペクトルディップとなる周波数 [Hz]				
Cmaj	440	720	860	930	……
Cmin	440	730	850	990	……
Cdim	430	690	880	990	……
Caug	460	590	710	890	……

そして，和音ごとにスペクトルディップが 9 もしくは 10 箇所あるので，入力のスペクトルにおいてディップリストで該当する周波数それぞれのパワーの平均をとる．さらにそれを今回候補とした和音全種のスコアの和で標準化し，それをスコア S_S(Score of Spectrum dip)とする (式 (3.1))．なお，このスコアは小さい方がよりその和音らしいものとする．Ps(Power of Spectrum dip)はディップリストで該当する周波数のパワーである．

$$S_S' = average(P_S)$$
$$S_S = \frac{S_S'}{\text{全和音の}S_S'\text{の合計}} \qquad (3.1)$$

3.3 クロマベクトル法

ここではもっとも単純な 12 次元クロマベクトルを用いる．クロマベクトルの作成方法は次のとおりである．各音名にあたる周波数の範囲を計算し，周波数の範囲のスペクトルに入ったスペクトルのパワーを合計する方法とし，範囲内のパワーの平均や補間などの操作は行わなかった．

クロマベクトルによる和音推定の方法は，対応する和音の構成音にあたるクロマベクトルのパワーP_C(Power of Chroma vector)の平均をとり，こちらも候補とした和音全種のスコアの和で標準化したものを各和音のスコア S_C(Score of Chroma vector)とした(式 (3.2))．こちらのスコアは大きい方がよりその和音らしいものとする．

$$S_c' = average(P_C)$$
$$S_C = \frac{S_C'}{\text{全和音の}S_C'\text{の合計}} \qquad (3.2)$$

3.4 スコアの統合

スペクトルディップ法のスコアが閾値より小さかった場合にはスペクトルディップ法による順位づけをそのまま採用し，大きかった場合にはクロマベクトル法のスコアと合わせそれぞれのスコアによる順位で重み付けを行った新たなスコアを算出する (式 3.4)．閾値は式 (3.3) のように決めた．なお，N は実験に用いる和音の種類数，a は定数でここでは 1/2 とした．Rs，Rc(Rank of Spectrum dip/Chroma vector)はそれぞれスペクトルディップ法，クロマベクトル法によるその和音の順位である．

$$\text{threshold} = \frac{1}{N} \times a \qquad (3.3)$$

$$S_m = \frac{S_c}{S_s \times 2^{R_s} \times 2^{R_c}} \qquad (3.4)$$

4. 評価実験

4.1 実験方法

実験は単和音のみの音源と楽曲音源の二通りで行った．どちらの音源も「mino 式 midi」（midi シーケンサー）で作成した midi ファイル（音色：ピアノ）を「Timidi95」（midi ファイルから wav ファイルへの変換ソフト）を用いて 44100Hz でサンプリングした wav ファイルに変換したものを使用した．変換の際にホワイトノイズが若干かかったが，それ以外には特にノイズなどは含まれない音源となっている．単位時間ごとの切り出しにはハミング窓を用いた．

楽曲音源の正解データのアノテーションは自らの手で行った。

なお、プログラムの実装は Matlab で行った．

4.1.1 単和音に対する実験

12 音それぞれに対して長三和音（maj），短三和音（min），増三和音（aug），減三和音（dim） の 4 種類の和音の基本形・第 1 転回形・第 2 転回形を実験の対象とした．重音の多少による違いも観察するため 3 重音と 6 重音の 2 パターンで実験を行った．和音の音長は各 2 秒，解析するフレームの長さは 0.1 秒とした．実験を行った和音の音高の範囲を maj の基本形の場合で示したものが図 4.1 である．

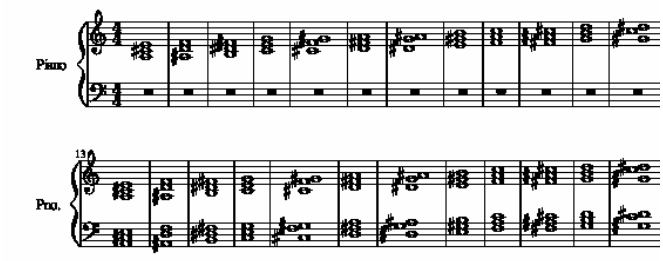

図 4.1 実験の対象とした和音の範囲

4.1.2 楽曲に対する実験

単和音での実験に対して，和音構成音以外の音が含まれる音源，つまり楽曲に対する実験も行った．メロディーラインなどがノイズとして解析に与える影響を調べるためである．こちらでは「慶應歌集」からピアノ伴奏が含まれている 29 曲を抜粋して実験を行った．解析のフレームの長さは単和音の場合と同じく 0.1 秒とした．

4.2 実験結果

4.2.1 予備実験

最適なフーリエ変換のポイント数を求めるために，まず予備実験を行った．Amaj の基本形・3 重音のデータを用い，フーリエ変換のポイント数を 512, 1024, 2048, 4096, 8192, 16384, 32768 としてスペクトルの解像度によって推定の精度がどう変化するのかを調べた．各フレームの長さにおける窓のシフト幅は，各フレームの長さの半分とした．

この結果から，計算量がなるべく小さくなり、かつ精度が落ちないという条件を満たすものとして，今回の実験では FFT のポイント数を 4096 にして実験を行うことにした．入力される音響信号は 44100Hz であるから，分解能は 44100/4096≒10.77Hz となる．

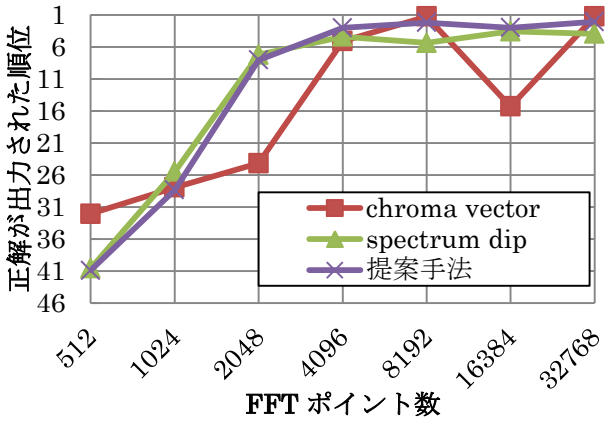

図 4.2 FFT のポイント数と推定精度の相関

4.2.2 単和音に対する実験

結果は正解の和音が何位に出力されたかという順位で出力される．以下に和音の種類毎に結果をまとめたものを記す．表中の値はフレームごとに正解が出力された順位の平均値であり，「CV」はクロマベクトル法単独の結果であることを，「SD」はスペクトルディップ法単独の結果であることを示す．各条件でもっとも結果が良かった手法のスコアを太字で示している．

表 4.1　各手法の単和音実験における平均正解順位

	CV	SD	提案手法
maj　3 重音	3.568	4.498	**2.982**
maj　6 重音	4.608	2.339	**2.100**
min　3 重音	3.779	6.760	**3.731**
min　6 重音	3.465	3.391	**2.278**
aug　3 重音	**3.391**	7.081	5.057
aug　6 重音	3.842	3.724	**3.151**
dim　3 重音	3.448	3.832	**2.509**
dim　6 重音	4.271	**1.909**	2.012
平均	3.796	4.192	**2.978**

4.2.3 楽曲に対する実験

結果はフレームごとに，単和音の場合と同様に正解の和音が何位に出力されたかで示される．全楽曲の平均はクロマベクトルが 15.18 位，スペクトルディップが 11.57 位，提案手法が 11.31 位となった．順位の平均では大きな差は見られなかったが，29 曲中 17 曲で提案手法が最良の結果を出すことができた．

4.3 考察

4.3.1 単和音

8 種の和音のうち 6 種で提案手法が最良の結果を出すなど，既存の手法に対して十分有効な結果を出すことができたと言えるだろう．この結果を出せた理由として次のことが考えられる．表 4.2 はスペクトルディップ法・クロマベ

クトル法・提案手法のそれぞれにおける Amaj の基本形・3 重音の場合の最初の 5 フレームの上位 10 候補をまとめたものである．これらを見ると，既存手法それぞれの実行結果は上位であってもかなり異なることが分かる．片方で上位にある候補でも，算出法が全く異なるためにもう一方ではかなり下位にある例もある．しかし提案手法の出力を見てみると両手法ともに『悪くない』順位であった候補が提案手法での上位に来ていることがわかる．このことが結果的に精度を高めることとなったと考えられる．

表 4.2　Amaj 基本形 3 重音の各手法による解析結果

フレーム	手法	出力									
		1 位	2 位	3 位	4 位	5 位	6 位	7 位	8 位	9 位	10 位
1	スペクトルディップ法	F#min	Bmaj	Dmaj	C#aug	Faug	Aaug	**Amaj**	Dmin	C#min	F#dim
	クロマベクトル法	Amin	A#dim	**Amaj**	Fmaj	A#min	Aaug	Faug	C#aug	A#maj	Dm
	提案手法	**Amaj**	C#aug	Faug	Aaug	Amin	Dm	Fmaj	A#dim	C#min	Adim
2	スペクトルディップ法	**Amaj**	C#aug	Faug	F#min	Aaug	Bmaj	Dm	Dmaj	C#min	F#dim
	クロマベクトル法	A#dim	A#min	**Amaj**	Amin	A#maj	Aaug	Faug	C#aug	Fmaj	Dmin
	提案手法	**Amaj**	C#aug	Faug	Aaug	Amin	Dmin	A#dim	C#min	Fmaj	A#min
3	スペクトルディップ法	F#min	**Amaj**	Aaug	C#aug	Faug	Bmaj	Dmaj	Dmin	C#min	F#dim
	クロマベクトル法	A#dim	A#maj	A#min	Amin	**Amaj**	Dmin	Adim	Fmaj	Aaug	Faug
	提案手法	**Amaj**	Aaug	C#aug	Faug	Dmin	Amin	A#dim	Fmaj	C#min	Adim
4	スペクトルディップ法	**Amaj**	F#min	Aaug	C#aug	Faug	Bmaj	Dmaj	Dmin	C#min	D#dim
	クロマベクトル法	A#dim	A#maj	A#min	**Amaj**	Amin	Dmin	Aaug	Faug	C#aug	Fmaj
	提案手法	**Amaj**	Aaug	C#aug	Faug	Dmin	Amin	A#dim	C#min	Fmaj	A#maj
5	スペクトルディップ法	Aaug	F#min	C#aug	Faug	**Amaj**	Bmaj	Dmin	C#min	Dmaj	Amin
	クロマベクトル法	A#dim	A#min	A#maj	**Amaj**	Aaug	Faug	C#aug	Dmin	Adim	Amin
	提案手法	Aaug	**Amaj**	C#aug	Faug	Dmin	A#dim	Am	C#min	Fmaj	A#min

4.3.2 楽曲

単和音では提案手法と既存手法の間に明確な差が出たが，楽曲の場合では最良の結果を出せたもののそこまで明瞭な差は出なかった．最良の結果が出た曲数が既存手法に比べ提案手法のほうが多くなったのは、単和音のときにも見られたように，スペクトルディップ法・クロマベクトル法双方の確信度が高い解答を採用しやすいことによると考えられる．

ここで特に結果が悪かった曲を見ると，良い結果を出せなかった理由として次の 3 点が考えられる．

まずメロディーラインとして和音の構成音でない音も音源に含まれていたこと．人間の耳はメロディーとそうでないものを聞き分けることができるが機械にはそれができない．全てを合わせて解析してしまうためメロディーラインはただのノイズと同じように和音の特徴を消してしまうことになる．そのため構成音でない音を含む音源はクロマベクトルもスペクトルディップも共に苦手としているのである．

次に，和音自体も分散和音として演奏されていることが多かったことが挙げられる．毎分 120 拍（BPM120）の曲の場合，今回実験に用いたフレーム長 0.1 秒は十六分音符

よりも短い長さとなる．これでは分散和音を和音として捉えることができない．だが分散和音を解析しやすくできると考え解析時のフレームを長くしても，結果は向上しなかったためこの点に関しては別途対策を立てる必要がある．

最後は低音が多く含まれていたことである．音高の周波数の問題から低音部ではどうしても FFT の精度が悪くなってしまう．具体的には，音名 C（中央ドより 2 オクターブ低いド）以下の音が含まれていると急激に精度が落ちることがわかった．

単和音での実験結果から、逆にいえばメロディーライン・分散和音・極端な低音が含まれていない楽曲，具体的に言えばアコースティックギターなどでコードのみを弾いているような場合であれば，4096 という少ないポイント数でも和音推定として実用に足る結果を出すことができているということになる．

4.3.3 実行時間

本研究ではリアルタイムでの実行を視野に入れ，推定の精度だけでなく短時間に実行できることも目標としていた．そこで，スペクトルディップ法の約 5 倍の計算時間を要するクロマベクトル法の計算を呼びだす回数を可能な限り少なくするために，提案手法では設定した条件を満たしたときのみクロマベクトル法のスコア算出を呼び出している．実際，プログラムの実行過程を解析するとクロマベクトル関数の呼び出し回数が半減している例も見受けられた．ところが，Matlab によるプログラムの内部処理方法によるものと思われるが，音源のデータ量によって CPU のクロック数が最悪の場合で 1/3 倍まで落ち，その結果命令量が半減しても実行時間が約 3 倍になっているものがあった．

これらの結果の一部をまとめたのが表 4.6 である．「**times」はその関数の実行回数を，「**sec」は実行時間（秒）を示す．この問題は正直想定していなかった．目標の一つとして実行時間の短縮も掲げていたが，実行時のスレッド数の管理までは本研究の目指すところではないため，この問題は今後の課題としたい．

表 4.5　クロマベクトル法を毎回実行した場合（全実行）と条件を満たした時のみ実行した場合（提案手法）のメモリのクロック数及びクロマベクトル法/スペクトルディップ法のプログラムの実行回数/処理時間の比較

曲 No.	曲の長さ(秒)		秒間クロック数	CV times	CV sec	SD times	SD sec	all sec
1	87.7	全実行	2.03E+09	1722	52.778	1722	10.369	80.588
		提案手法	3.03E+09	1464	139.885	1722	32.578	217.401
		増減	+49.18%	−14.98%	+165.05%	0.00%	+214.18%	+169.77%
2	48.0	全実行	3.03E+09	957	257.450	957	55.442	393.333
		提案手法	2.03E+09	842	69.889	957	16.148	107.041
		増減	−32.97%	−12.02%	−72.85%	0.00%	−70.87%	−72.79%
3	42.0	全実行	2.03E+09	813	59.246	813	11.871	89.717
		提案手法	2.03E+09	431	33.620	813	12.257	61.857
		増減	0.00%	−46.99%	−43.25%	0.00%	+3.25%	−31.05%
4	72.0	全実行	2.03E+09	1443	98.142	1443	19.142	150.961
		提案手法	4.05E+09	727	36.138	1443	13.871	69.243
		増減	+99.18%	−49.62%	−63.18%	0.00%	−27.54%	−54.13%
5	64.0	全実行	1.00E+09	1293	31.455	1293	6.910	49.109
		提案手法	4.05E+09	1270	64.622	1293	14.092	96.300
		増減	+305.00%	−1.78%	+105.44%	0.00%	+103.93%	+96.09%

5. おわりに

以上で述べた通り，本研究で提案した手法は既存の 2 手法に比べ，単和音・楽曲ともにより優れた精度で和音推定を行うことができた．特に単和音においては多重音の場合でもそうでなくても従来手法よりも高い精度を得られた．

今回提案した手法は 2 つの既存手法を組み合わせるものだったため，推定精度の面での改良点として次の 3 点が挙げられる．第一は使用している既存手法それぞれの精度を上げること．第二はスコアを併合する際に用いる式を改良すること．この式は様々な式を試して辿り着いたものではあるが，改良の余地はまだあるだろう．そして第三はより多くの手法を組み合わせるものである。実行時間は増えると考えられるが，合議制を採ることで更なる精度の向上が見込まれる．また，和音の種類は本来 maj・min・aug・dim の 4 種に留まらずセブンス 7 種など多岐に渡っているため，今後は対応する和音をさらに増やしていく必要がある．他方，実行時間の短縮を図るためには，和音進行を学習させたシステムと組み合わせ和音候補の枝切りを行う・実装方法を再検討するなどの方法が考えられる．

参考文献

[1] 須見康平,糸山克寿,吉井和佳,駒谷和範,尾形哲也,奥乃博. ベース音高と和音特徴の統合に基づく和音系列認識. 情報処理学会論文誌. Vol. 52, No.4. 2011

[2] Mark A. Bartsch and Gregory H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. IEEE TRANSACTIONS ON MULTIMEDIA, Vol. 7, No. 1, 2005.

[3] J.P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In Proceedings of the International Symposium on Music Information Retrieval, 2005.

[4] T. Cho and J.P. Bello. REAL-TIME IMPLEMENTATION OF HMM-BASED CHORD ESTIMATION IN MUSICAL AUDIO. In Proceedings of the International Computer Music Conference, pp. 117–120, 2009.

[5] A. Sheh and D.P.W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In Proceedings of the International Symposium on Music Information Retrieval, pp. 185–191, 2003.

[6] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. IEEE Transactions on Audio Speech and Language Processing, Vol. 16, No. 2, p. 291, 2008.

[7] 藏内雄貴, 松原正樹, 大野将樹, 斎藤博昭. 周波数スペクトルの谷状点系列による和音推定. 情報処理学会 音楽情報科学研究会 研究報告 2008-MUS-76-21, 第 2008 巻, pp. 125‐130, 2008.

[8] 宮田聡, 上野佑馬, 藏内雄貴, 松原正樹, 斎藤博昭. スペクトルディップ法による和音推定. 第 8 回情報科学技術フォーラム (FIT 2009), No. E-036, 2009.

[9] 藏内雄貴. スペクトルディップ法を用いた楽曲の和音推定, 慶應義塾大学理工学研究科修士論文,2010.