卒業論文

くし形フィルタと自己相関関数による演奏楽器推定

東海林研究室

5 年　情報工学科　13 番

北見伸一郎

目次

1 章　はじめに

楽曲の演奏音声から楽譜を作成することを採譜という．近年，それを自動でおこなう自動採譜システムの研究が盛んにおこなわれている[1].

自動採譜システムの構築においては，音高，音長，拍子等の推定が必要であるが，それと同時に演奏楽器の推定もおこなう必要がある．

現在，楽器推定法では，各楽器の音高ごとのスペクトルをテンプレートとして記憶しておき，演奏楽器のスペクトルを測定して，そのテンプレートマッチングにより楽器を推定するものが提案されている[2]．この方法では対象とする楽器や演奏形態の増加に伴ってテンプレートの数とファイルのサイズも増え，テンプレートマッチングにも時間がかかるという問題がある．

そこで本論文では，くし形フィルタを用いることによりテンプレートの数を減らし，さらに自己相関関数を用いてテンプレートマッチングをおこなうことでファイルのサイズを減らす演奏楽器推定システムの構築を検討する．

本論文の構成は以下のとおりである．

1 章では本論文の背景，目的，全体の内容の説明をおこなっている．

2 章ではくし形フィルタについての説明をおこなっている．まず，音楽についての説明でオクターブや音名，楽器音と倍音特性の関係の説明をおこない，それから，くし形フィルタの原理と性質の説明，さらに，くし形フィルタの利得の導出過程を説明している．

3 章では，ファイルサイズを減らすために用いられる自己相関関数について説明をおこなう．また自己相関関数とパワースペクトルとの関係も説明している．

4 章ではテンプレートマッチングの説明をおこなう．

5 章では具体的な演奏楽器推定方法について説明をおこなう．

6 章ではテンプレートの作り方の説明をおこなう．またテンプレート間の相関についての考察をおこなう．

7 章ではテンプレートに関する実験の条件・結果・考察を書いている．

8 章では単音での演奏楽器推定の条件・結果・考察を書いている．

9 章では単音（長時間）での演奏楽器推定の条件・結果・考察を書いている.

10 章では 2 和音での演奏楽器推定の条件・結果・考察を書いている.

11 章では本論文のまとめをおこなっている.

2 章　くし形フィルタについて

2-1 オクターブと音名について

音名は日本語の五十音や英語のアルファベットと同じように, 1 つ 1 つの音につけられた名前である[3]．本論文では英語の読み方を使って音名を表す（表 1)．

また C から C，A から A，のように一回り上または下の同じ音までの間を 1 オクターブという[3]．ピアノの 1 番左（低い音）のオクターブはオクターブ 1 となりピアノの中央のオクターブはオクターブ 4 とる．

表 1：音名の表

イタリア語	ド	ド#	レ	レ#	ミ	ファ	ファ#	ソ	ソ#	ラ	ラ#	シ
英語	C	C#	D	D#	E	F	F#	G	G#	A	A#	B

2-2 楽器音と倍音特性

音高は基本周波数によって決まる[4]．また，音の周波数が 2 倍になるとオクターブが 1 つあがる．しかし，音高が同じでも楽器により音色が異なるのは楽器により倍音成分のスペクトルが違うからである．音色は基本周波数とその倍音成分により決定されている．図 1 を見てわかるように倍音成分は基本周波数の整数倍の周波数であり楽器によりそのスペクトルが異なる．つまり同じ音高でも楽器によって音色が変わるのは，倍音成分の違いによるものである．

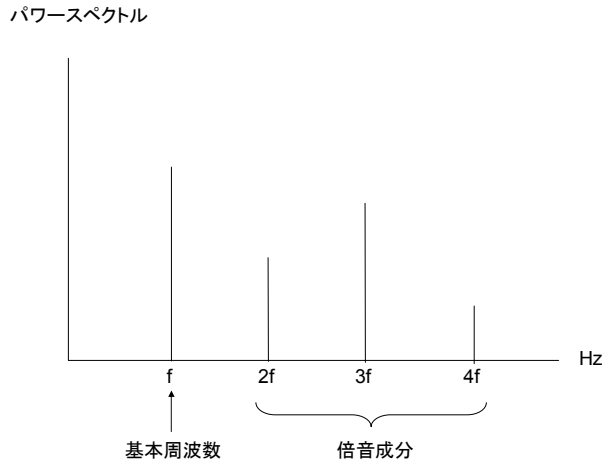


図 1：音色の構成

2-3　くし形フィルタの原理と性質

くし形フィルタには Notch 型くし形フィルタと Resonator 型くし形フィルタがある[1]. Notch 型くし形フィルタは基本周波数とその倍音成分だけを除去する特性を持っていて，逆に Resonator 型くし形フィルタは基本周波数とその倍音成分の周辺だけを残し，他の部分を減衰させる特性を持っている．本研究ではくし形フィルタの出力を使って実験するので，Resonator 型くし形フィルタを用いる．

整数 i をオクターブ，整数 p を音名番号（p = 1,2,…,12，がそれぞれ C,C#,…,B に対応）とすると，任意の音は i と p の組（i,p）で表される．例えば（3,1）は O3C（オクターブ 3 の C），（4,2）は O4C#に対応する．

また $f_{i,p}$(Hz)を（i,p）音の基本周波数とし，f_s(Hz)をサンプリング周波数とする．このとき（i,p）音に対応する正整数の定数として

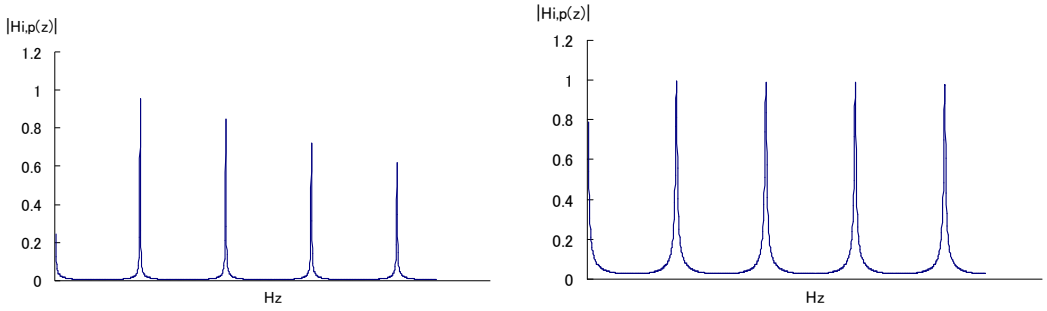
$$N_{i,p} = [f_s / f_{i,p}] \quad ([\,]：整数への端数処理) \qquad (1)$$

を定義すると，(i,p) 音に対応する Resonator 型くし形フィルタは次の伝達関数で表される[1].

$$H_{i,p}(Z) = \frac{1-a}{1-a \cdot z^{-N_{i,p}}} \qquad (2)$$

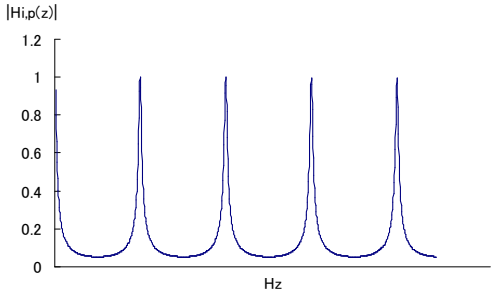
このフィルタは出力 $y_{i,p}(n)$を $N_{i,p}$ 時刻だけ遅延させて入力に加算するだけの単純な IIR 型フィルタであり，利得特性 $|H_{i,p}|$ は図 2 で表される．図 2 から，Resonator 型くし形フィルタは（i,p）音に含まれる基本周波数とその倍音成分の周辺を残し，その他の成分を減衰させる特性を持つことがわかる．つまり楽器の特徴をうまく取り出すことが出来る．なおフィルタ係数 0≦a＜1 は任意のパラメータであり，a が 0 に近づくほど利得は平坦になり，1 に近づくほど急になる．

この特性より（i,p）音に対応するくし形フィルタ $H_{i,p}(z)$に任意の和音を通して得た出力 $y_{i,p}(n)$には（i,p）音の成分が多く含まれている．つまり，和音を単音に分離できるので，和音を分離しないで全ての楽器音の組み合わせに対してテンプレートマッチングをおこなうよりも，くし形フィルタから出力された単一楽器音に対してテンプレートマッチングをおこなうことでテンプレート数を減らすことが出来る．例えば，オクターブ 3～5 で 2 和音のとき，和音を分離しないと 36×35 個のテンプレートが必要となるが，分離すると 36 個のテンプレートですむ.

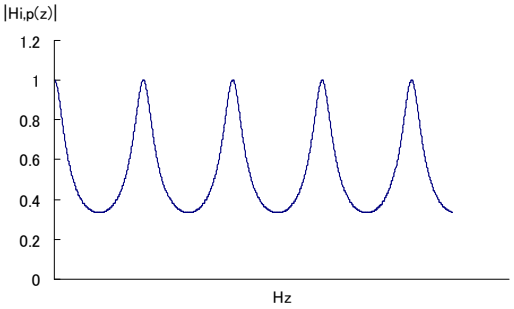


(a = 0.99)

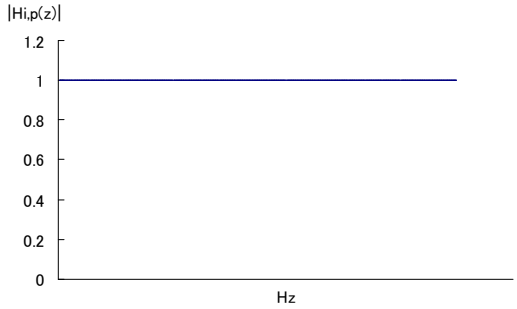
(a = 0.95)



(a = 0.9)



(a = 0.5)



(a = 0)

図 2：Resonator 型くし形フィルタの利得

2-4　くし形フィルタの利得の導出

ここで(2)式より図 2 のような利得が得られることを説明する.

まず, フーリエ変換とラプラス変換及び z 変換について説明する. フーリエ変換とは時間領域の関数 f(t)を周波数領域の関数 F(ω)に変換するもので, 式で表すと

$$F(\omega) = \int_0^\infty f(t) \cdot e^{-jwt} dt \qquad (3)$$

となる.

しかし，$T \to \infty$ のとき f(t)が収束せずフーリエ変換の値が∞になるときがある．
そこで $e^{-\sigma t}$ を掛けて $f(t) \cdot e^{-\sigma t}$ を収束させる．
これを式で表すと

$$\lim_{t \to \infty} \left| f(t) \cdot e^{-\sigma t} \right| = 0 \qquad (4)$$

となる．これをさらにフーリエ変換させると

$$\int_0^{\infty} f(t) \cdot e^{-\sigma t} \cdot e^{-j\omega t} dt \qquad (5)$$

となり，s＝ σ＋ωt と置くと

$$F(s) = \int_0^{\infty} f(t) \cdot e^{-st} dt \qquad (6)$$

となる．これがラプラス変換である．Z 変換は離散時間信号 $f_s(t)$をラプラス変換し，e^{st} を z と置き換えたものであり次のように定義される

ここから(2)式を考える．(2)式はくし形フィルタのインパルス応答 h[k]の z 変換である．まず，(2)式の z を e^{st} に置き換えて、h[k]の z 変換から h[k]のラプラス変換にする．すると h(k)のラプラス変換は(7)式で表される．

$$H(s) = \frac{1-a}{1-a \cdot e^{-stN}} \qquad (7)$$

$s = \sigma + j\omega$ とする．

次に，$\sigma = 0$ を代入することで(2)式を h(k)のフーリエ変換にする．(8)式と表される．

$$H(\omega t) = \frac{1-a}{1-a \cdot e^{-j\omega tN}} \qquad (8)$$

(8)式に絶対値をとることで利得となる．

$$\left| H(\omega t) \right| = \left| \frac{1-a}{1-a \cdot e^{-j\omega tN}} \right| \qquad (9)$$

(9)式を計算する．オイラーの公式より

$$\left| H(\omega t) \right| = 1-a \Big/ \sqrt{(1-a\cos\frac{\omega}{f_s}N)^2 + a^2 \sin^2 \frac{\omega}{f_s}N} \qquad (10)$$

$$\left| H(\omega t) \right| = 1-a \Big/ \sqrt{1-2a\cos\frac{\omega}{f_s}N + a^2} \qquad (11)$$

fs：サンプリング周波数

となり，図 2 のような利得が得られる．

3 章　自己相関関数について

自己相関関数とは，１つの信号波形中に繰り返し（周期性）があるかどうかを調べるものである[5]．自己相関関数を求めることによりその信号波形がどのような周期性を持っているか（いないか）を知ることが出来る．また，１周期を構成している時間やデータの個数，振幅の大きさがわかる．具体的にサンプル数 M からなる信号 x(n)を考えると，

（ｉ） X(n)に対して，x(n)を一定時間 m だけずらして x(n+m)とする．
（ii） X(n)と x(n+m)の積を各時刻において求める．
（iii） それらを加え合わせ，平均を求める．

と操作することによって自己相関を求めることが出来る[6]．これらのことから循環型の自己相関の式は(12)のように表され，非循環型の自己相関の式は(13)のように表される．

$$\phi_{xx}(m) = \frac{1}{M}\sum_{n=1}^{M} x(n)x((n+m) \bmod M) \qquad (12)$$

$$\phi_{xx}(m) = \frac{1}{M}\sum_{n=1}^{M} x(n)x(n+m)\text{，（ if （n+m）< M ）} \qquad (13)$$

なお，ある信号波形の自己相関関数をフーリエ変換するとパワースペクトルが得られ，逆にパワースペクトルをフーリエ逆変換することにより，自己相関関数を求めることが出来るという関係がある（図 3）．

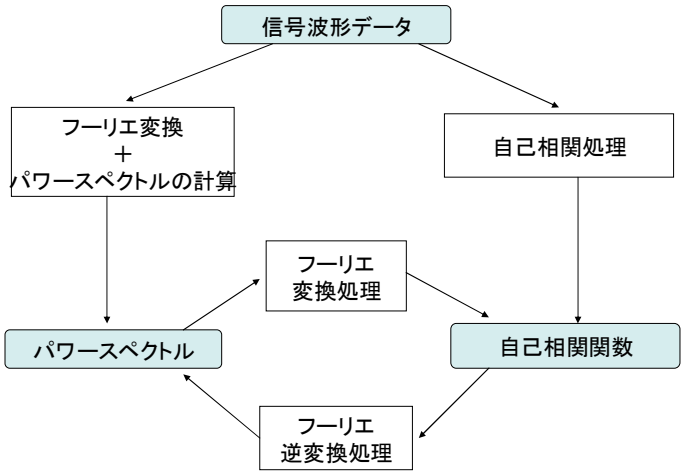


図 3：自己相関とパワースペクトルの関係

4 章　テンプレートマッチングについて

テンプレートマッチングとは，抽出したスペクトルなどの特徴量を，あらかじめ知られている代表的な特徴（テンプレート）との相関係数を比較することである[7].

例えば、図 4 のような抽出されたグラフがあったとする．その抽出されたグラフとテンプレート 1，テンプレート 2，テンプレート 3 をそれぞれ比較していく．この場合，最も似ているのはテンプレート 2 なのでテンプレート 2 との相関係数が 1 番高くなる．

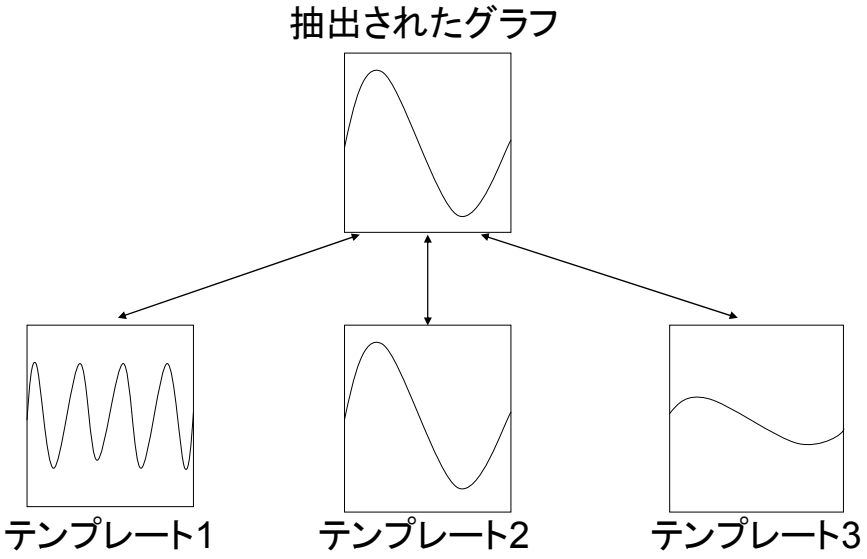


図 4：テンプレートマッチングの例

入力信号を f，テンプレートを t，データ長を n とすると相関係数の式は(14)のようになる．

$$R = \frac{\sum_{k=0}^{n-1}(f[k]-\overline{f})(t[k]-\bar{t})}{\sqrt{\sum_{k=0}^{n-1}(f[k]-\overline{f})^2}\sqrt{\sum_{k=0}^{n-1}(t[k]-\bar{t})^2}} \qquad (14)$$

5 章　演奏楽器推定システムの構成

本研究で提案する演奏楽器推定システムは次の手順で動作する．ここで時刻 n におけるシステムへの入力音声信号を $x(n)$とし，(i,p)音に対応するくし形フィルタを $H_{i,p}(z)$とする．

（ｉ） オクターブ i の範囲を決定し，全ての（i,p）の組に対応した $H_{i,p}(z)$を作成してそれらを並列接続する（図 5）．時刻 n ごとに $x(n)$を並列接続された全ての $H_{i,p}(z)$に入力し，それぞれのフィルタ出力を得る．

（ⅱ） ある時間間隔ごとに，全ての $y_{i,p}(n)$に対する短時間での標準偏差 $s_{i,p}$ を求める．

(ⅲ) 今回は音高推定が目的ではないので，オクターブの違いを無視することにして各 p 音に対して標準偏差の和 $s_p=\Sigma_i s_{i,p}$, p = 1,2,…,12 を求める．ここで，ある s_p が任意に定めたしきい値 TH_S よりも大きい場合は，あるオクターブにおいて p 音が演奏されていると判断することが出来る．

(ⅳ) 演奏されていると判断した p 音に対して，各オクターブ別に $y_{i,p}(n)$から自己相関関数 $r_{i,p}(k)$,k = 1,2,…,Rmax を求めてテンプレートマッチングをおこなう．ここで Rmax は自己相関関数長である．今回はテンプレートマッチングとして $r_{i,p}(k)$とテンプレートとの相関を使用するので，マッチングの指標を $R_{i,p}(l)$ （l:楽器番号）とおくと，$R_{i,p}(l)$の最大値は 1，最小値は-1 となる．さらに $l' = \mathrm{argmax}_l R_{i,p}(l)$とすると，もし $R_{i,p}(l')$が任意に定めたしきい値 TH_R より大きい場合は p 音は楽器 l'で演奏されていると判定できるので楽器番号 l'を記録する．

(ⅴ) (ⅲ)と(ⅳ)を p = 1,2,…,12 に対して繰り返す．

(ⅵ) 短時間で楽器が頻繁に切り替わることはあまり考えられないので，楽曲のフレーズ単位で楽器認識をおこなう．フレーズ終了時に記録した楽器番号の頻度を計算し，頻度が大きい楽器の順に楽器名を示す．

なお（ⅳ）において FFT ではなく自己相関関数をもちいてテンプレートマッチングをおこなっている理由は，特に中低音において十分な周波数分解能を得るために，自己相関関数よりも FFT の方が超時間の音声信号が必要だからである[8]．例えば，サンプリング周波数を 44100[Hz]とすればサンプリング間隔は 1/44100 秒となる．自己相関長を 512 とすれば最も長い波の周期は 512/44100 秒であり，周波数は f=1/T の式より f≒86.13[Hz]となる．同様に 2 番目に長い波の周波数は 86.30[Hz]なので，周波数分解能は約 0.17[Hz]となる．一

方 FFT の周波数分解能は一様に 86[Hz]である．なお本論文では O3C（130[Hz]）より低い音は使わないので自己相関長が 512 であれば十分な周波数分解能が得られる．

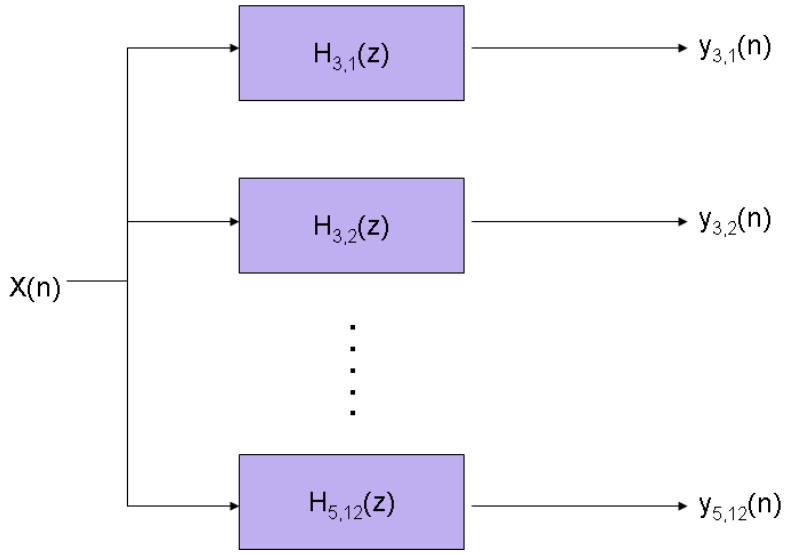


図 5：くし形フィルタの並列接続

6　テンプレートの作成

6-1　テンプレートの作り方

提案する演奏楽器推定システムでは，入力音声信号はくし形フィルタに通されて利得の影響を受けるため，通常の楽器入力音から自己相関関数を求めてテンプレートとして用いることは出来ない．そこで楽器の（i,p）音をくし形フィルタ $H_{i,p}(z)$に通して求めた自己相関関数を（i,p）音に対するテンプレートとして用いることにする（図 6）．以上のことから本実験における自己相関長はテンプレート長と一致する．

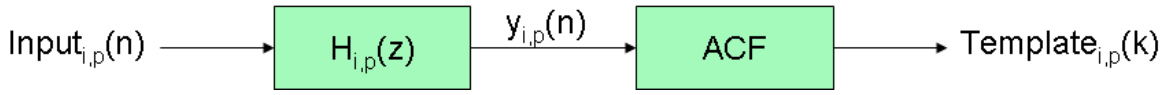


図 6：テンプレートの作成方法

6-2　テンプレート間の相関

同音高でのテンプレートの相関を見たとき，テンプレート間の相関が高ければそのテンプレートとそのテンプレートは似ていることを意味する．逆にテンプレート間の相関が低ければ，そのテンプレートとテンプレートは似ていないことを意味する．つまり，テンプレート間の相関が高ければ，楽器推定のときその楽器と相関の高い楽器の区別がつけられず間違えて推定されやすくなり，テンプレート間の相関が低ければ，楽器推定のとき区別がつけられ他の楽器と間違えて推定されることが少なくなる．

このことから，テンプレート間の相関が低くなるようにくし形フィルタのフィルタ係数 a の値や自己相関長を決定する必要がある．

7 章　実験 1（テンプレート間の相関に関する実験）

7-1　　実験概要

はじめにテンプレート間の相関を低くする実験をおこなった.

音声信号は MIDI 出力を変換して作成した wav ファイル（44.1Hz, 16bit, モノラル, 2 秒）とした. 対象楽器はピアノ（PF, GM=1), トランペット（TR, GM=57), アルトサックス（AS, GM=66), バイオリン（VN, GM=41), フルート（FL, GM=74), コントラベース（CB, GM=44), の 6 種類とし, 楽器のチューニングで用いられる O4A（基本周波数 440[Hz]）の定常状態にある音を用いた.

これらの各楽器音からテンプレートを作成しテンプレート間の相関を求めた.

7-2　結果

テンプレートを作る際, テンプレートの長さ（自己相関長）を 512, 1026 と変え, くし形フィルタの係数 a の値を 0.9, 0.95, 0.99 と変え, さらに自己相関を循環させる場合とさせない場合で計算し, 相関の平均が 1 番低くなる条件を検討した. その結果を表 2 に示す.

結果は, くし形フィルタの係数 a=0.95, テンプレートの長さ 512, 循環自己相関で計算したものが, テンプレート間の相関の平均が 1 番低かった. 表 3 はその条件でのテンプレート間の相関で, 背景が灰色になっているのは相関が 0.7 以上のところである. また, 図 7 は各テンプレートをグラフにしたものである.

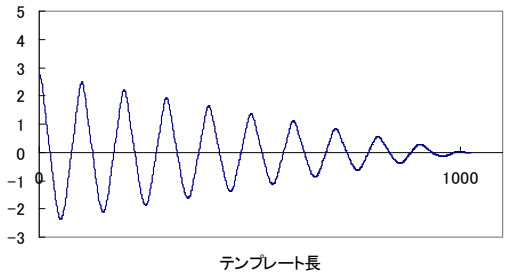
表 2：各条件での相関の平均

テンプレート長	フィルタ係数a	循環/非循環	相関の平均
512	0.9	循環	0.632
512	0.9	非循環	0.638
512	0.95	循環	0.599
512	0.95	非循環	0.633
512	0.99	循環	0.619
512	0.99	非循環	0.725
1024	0.9	循環	0.633
1024	0.9	非循環	0.634
1024	0.95	循環	0.618
1024	0.95	非循環	0.629
1024	0.99	循環	0.727
1024	0.99	非循環	0.726

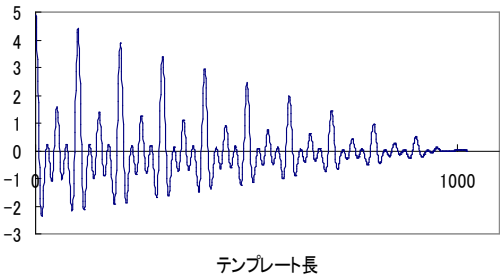
表 3：テンプレート間の相関

（a=0.95，テンプレートの長さ 512，非循環自己相関で計算）

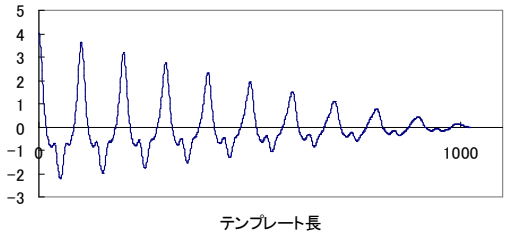
	PF	TR	AS	VN	FL	CB
PF	1.000	0.355	0.972	0.668	0.010	0.981
TR	0.355	1.000	0.456	0.603	0.752	0.466
AS	0.972	0.456	1.000	0.675	0.043	0.997
VN	0.668	0.603	0.675	1.000	0.041	0.669
FL	0.010	0.752	0.043	0.041	1.000	0.087
CB	0.981	0.466	0.997	0.669	0.087	1.000



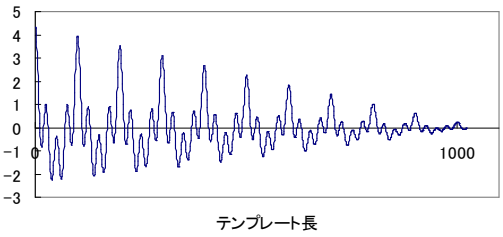
PF のグラフ



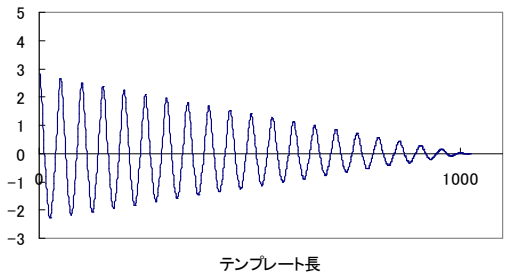
TR のグラフ



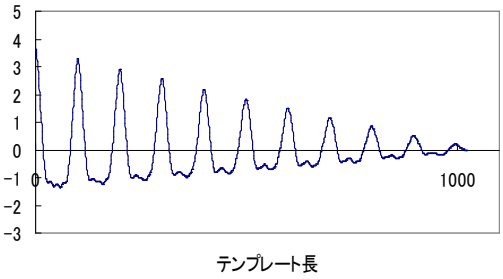
AS のグラフ



VN のグラフ



FL のグラフ



CB のグラフ

図 7：各テンプレートのグラフ

7-3　考察

表 3 からわかるように，PF と AS，PF と CB，AS と CB の相関が高い．これは図 7 の PF，AS，CB の各グラフを見てもわかるように似たグラフになっている．

この実験では電子音を用いての実験だったので周波数や音の強弱等の違いがなく機械的に正確なものだった．しかし，実際の楽器音を用いると周波数や音の強弱等に若干の違いが出る可能性があるので，実際の楽器音を用いたときテンプレート間の相関が低くなるかもしれない．

8 章　実験 2（単音での演奏楽器推定）

8-1　実験概要

電子音（MIDI）を用いて提案する演奏楽器推定システムの実験をおこなった.

入力音声信号はMIDI出力を変換して作成したwavファイル(44.1Hz, 16bit, モノラル, 2秒）とした．対象楽器はピアノ（PF，GM=1)，トランペット（TR，GM=57)，アルトサックス（AS，GM=66)，バイオリン（VN，GM=41)，フルート（FL，GM=74)，コントラベース（CB，GM=44)，の 6 種類とし，楽器のチューニングで用いられる O4A（基本周波数 440[Hz]）の定常状態にある音を用いた.

テンプレートを作成する際，7 章での実験結果をもとに，テンプレート長（自己相関長）を 512，またくし形フィルタのフィルタ係数 a の値を 0.95 とし，循環自己相関を用いてテンプレートを作成した.

また，テンプレートマッチングをおこなうタイミングは 100 ミリ秒おきとし，TH_S をそれぞれ表 4 のように設定し，$TH_R = 0.8$ と設定した．入力音声信号は，単音 PF，TR，AS，VN，FL，CB のそれぞれのワンフレーズとした.

表 4：TH_S の値

	PF	TR	AS	VN	FL	CB
THsの値	300	1500	2000	200	2000	2800

8-2　結果

それぞれの音声を入力したときの楽器の判定頻度を調べた. 結果は表 5 のようになった. 括弧の中はテンプレートマッチングをおこなった際の相関の平均である.

表 5：各楽器の頻度（単位：%）

入力音声＼テンプレート	PF	TR	AS	VN	FL	CB
PF	100 (0.999)	0 (0.577)	0 (0.973)	0 (0.729)	0 (0.152)	0 (0.980)
TR	0 (0.354)	100 (0.855)	0 (0.488)	0 (0.436)	0 (0.642)	0 (0.472)
AS	0 (0.912)	0 (0.674)	0 (0.931)	0 (0.714)	0 (0.353)	100 (0.944)
VN	55 (0.944)	0 (0.572)	0 (0.933)	10 (0.714)	0 (0.166)	35 (0.945)
FL	0 (0.058)	0 (0.768)	0 (0.116)	0 (0.062)	100 (0.999)	0 (0.087)
CB	0 (0.854)	0 (0.724)	0 (0.860)	0 (0.605)	0 (0.517)	100 (0.886)

8-3　考察

以上の結果から，PF，TR，FL，CB は正確に楽器推定できることがわかる．しかし，AS と VN の判定頻度が低いため，この 2 つを推定するのが困難なのがわかる．AS と CB は 7 章の実験結果からもわかるように，相関が非常に高い．AS を入力するとき CB を除くとうまく判定されると思われる．だが，VN と PF はテンプレート間の相関は高くないが，今回おこなった実験では相関が高くなっており，しかも VN と VN の相関が低くなっている．これは VN のテンプレートをとる位置が悪かったためと思われる．したがってテンプレートを最適な位置でとる必要があると思われる．また，しきい値の値を最適な値にする必要もある．

9 章　実験 3（単音での演奏楽器推定）

9-1　実験概要

電子音（MIDI）を用いて提案する演奏楽器推定システムの実験をおこなった.

対象楽器はピアノ（PF，GM=1），トランペット（TR，GM=57），アルトサックス（AS，GM=66），バイオリン（VN，GM=41），フルート（FL，GM=74），コントラベース（CB，GM=44），の 6 種類とした.

入力音声信号はMIDI出力を変換して作成したwavファイル(44.1Hz, 16bit, モノラル, 24 秒）で，全対象楽器の O4C～O5C とした.

テンプレートは実験 8 と同じものを使用した.

また，テンプレートマッチングをおこなうタイミングは 100 ミリ秒おきとし，TH_S=2000（VN だけ TH_S=600），TH_R = 0.8 と設定した.

9-2　結果

それぞれの音声を入力したときの楽器の判定頻度を調べた. 結果は表 6 のようになった.

表 6：実験結果（単位%）

入力音声＼判定楽器	PF	TR	AS	VN	FL	CB
PF	84.21	0	0	15.79	0	0
TR	0	81.25	0	0	18.06	0.69
AS	3.66	1.57	60.21	12.04	1.05	21.47
VN	24.1	0	12.01	45.78	0	18.07
FL	0	0	0	0	100	0
CB	6.63	0	13.27	9.69	0	70.41

9-3 考察

うまく推定できた．O4C～O5C を入力したのでその分正確に推定された部分が多くなったからだと思われる.

10 章　実験 4（2 和音での演奏楽器推定）

10-1　実験概要

電子音（MIDI）を用いて提案する演奏楽器推定システムの実験をおこなった.

対象楽器はピアノ（PF，GM=1），トランペット（TR，GM=57），アルトサックス（AS，GM=66），バイオリン（VN，GM=41），フルート（FL，GM=74），コントラベース（CB，GM=44），の 6 種類とした.

入力音声信号は MIDI 出力を変換して作成した wav ファイル（44.1Hz, 16bit, モノラル, 24 秒）で，対象楽器の全ての 2 和音の組み合わせの O4C～O5C とした.

テンプレートは実験 8 と同じものを使用した.

また，テンプレートマッチングをおこなうタイミングは 100 ミリ秒おきとし，TH_S を表 7 のようにし，$TH_R = 0.8$ と設定した.

表 7：TH_S の値

入力音声 ＼ 入力音声	PF	TR	AS	VN	FL	CB
PF		4800	8500	9300	3000	4000
TR			8000	9500	9400	8500
AS				10100	8200	9300
VN					8000	9500
FL						8700
CB						

10-2　結果

判定されるべき楽器の判定頻度が 1 位と 2 位なら A，判定されるべき楽器の判定頻度が両方とも 3 位以内なら B, 判定されるべき楽器の判定頻度が 1 楽器だけ 1 位か 2 位でもう 1 楽器が 4 位以下なら C，その他は D とした．結果は表 8 のようになった.

表 8：実験 4 結果

入力音声 / 入力音声	PF	TR	AS	VN	FL	CB
PF		B	A	A	A	A
TR			A	B	B	C
AS				A	C	A
VN					C	A
FL						A
CB						

10-3　考察

表 8 を見ると高確率で楽器推定できているのがわかる．A にならなかった楽器の組はテンプレートが似ているものがあり，その似ているテンプレートに誤判定されているものと思われる．似ているものはしょうがないが，テンプレートを最適な位置で取ることにより誤判定率は下がると思われる．

11 まとめ

本論文では，楽器推定をおこなうシステムを作った．従来の楽器推定システムはテンプレートファイル数とファイルサイズが膨大となるため，本論文ではくし形フィルタを用いることによりテンプレートの数を減らし，さらに自己相関関数を用いてテンプレートマッチングをおこなうことでファイルのサイズを減らすシステムの構築を検討した．

本論文で提案する楽器推定システムは, 入力音声を Resonator 型くし形フィルタに通し，その出力に対してさらに自己相関を計算する．そうして得られた信号と，あらかじめテンプレート音声を Resonator 型くし形フィルタに通して，その出力の自己相関を計算して得たテンプレートとの間でテンプレートマッチングをおこない，相関が高くなった楽器の順に演奏中の楽器として選ぶシステムである．

また本論文ではチューニングにも用いられる基本的な音である O4A を用いて実験をおこなった．はじめに，テンプレート間の相関に関する実験をおこなった．テンプレートを作る際, テンプレートの長さを 512, 1026 と変え, くし形フィルタの係数 a の値を 0.9, 0.95, 0.99 と変え，自己相関を循環させるのとさせないので計算し，相関の平均が 1 番低くなるものを検討した．結果は，くし形フィルタの係数 a=0.95，テンプレートの長さ 512，循環自己相関で計算したものが，テンプレート間の相関の平均が 1 番低かった．次に，テンプレート間の相関に関する実験の結果をもとに提案する演奏楽器推定システムの実験をおこなった．ピアノ，トランペット，フルート，コントラバスは正確に楽器推定できることがわかった．しかし，アルトサックスとバイオリンの楽器推定は良くなかった．

今回おこなった実験は電子音での実験だったが，より実用的にするために実際の楽器音を用いた実験が必要である．実際の楽器音で同じ実験をすると，今回の実験とは結果が変わってくると思われる．さらに今回は単音での実験だったが，多くの楽曲は複数楽器での演奏をおこなっている．よって，演奏楽器数を増やしての実験や，演奏楽器の組み合わせを変えた実験もおこなう必要がある．

謝辞

本研究は，豊橋技術科学大学高専教育連携プロジェクト「採譜システムにおける楽器推定法に関する研究」の援助によりおこなわれた.

参考文献

[1] 藤原道，山口満，斎藤努，田所嘉昭，Resonator 型くし形フィルタによる打楽器音を含む楽器音の音高推定法の検討，電子情報通信学会技術研究報告，pp.19-23，EA2003-80，2003.

[2] 三輪多恵子，田所嘉昭，斎藤努，くし形フィルタを利用した採譜のための異楽器音中のピッチ推定．電子情報通信学会論文誌，D-Ⅱ，Vol.J81-D-Ⅱ，No.9，pp.1965-1974，1998 年 9 月．

[3] 水谷友香，キッシー岸田，CD で覚えるやさしい楽譜の読み方，成美堂出版，2005.

[4] 春日正男，船田哲男，林伸二，武田一哉，音声情報処理，コロナ社，2001.

[5] 坂巻佳壽美，見てわかるディジタル信号処理，工業調査会，1998.

[6] 萩原将文，ディジタル信号処理，森北出版，2001.

[7] 酒井幸市，ディジタル画像処理入門，コロナ社，1997.

[8] 瀬戸康裕，夏井雅典，田所嘉昭，くし形フィルタと相関関数による音高推定困難和音の音高推定法，情報処理学会研究報告，pp.1-6，2006-MUS-68，2006.

[9] 小畑秀文，幹康，CAI ディジタル信号処理，コロナ社，1998.

[10] 坂内秀幸，夏井雅典，田所嘉昭，くし形フィルタに基づく自動採譜システムの実現，情報処理学会研究報告，pp.13-18，2007-MUS-71，2007.

[11] 酒井幸市，高専生のためのディジタル信号処理，コロナ社，1996.

[12] 大蔵康義，目で見る楽器の音 by FFT Analysis，株式会社国書刊行会，2004.

[13] 城戸健一，ディジタルフーリエ解析(Ⅰ) - 基礎編 - ，コロナ社，2007.

[14] 城戸健一，ディジタルフーリエ解析(Ⅱ) - 上級編 - ，コロナ社，2007.

[15] 三谷政昭，信号解析のための数学，森北出版，1998.

付録（ソース）

[テンプレートマッチング]

```
 double patan(double f[512],double t[512],int N)
{
	int	k;
	double	f_ave,t_ave,R,f_sum=0,t_sum=0;

	//合計
	for(k=0;k<N;k++)
	{
		f_sum += f[k];
		t_sum += t[k];
	}

	//平均
	f_ave = f_sum/N;
	t_ave = t_sum/N;

	double	r_bunsi=0,r_bunbo=0,r_bunbo_1=0;
	for(k=0;k<N;k++)
	{
		r_bunsi += (f[k]-f_ave)*(t[k]-t_ave);
		r_bunbo += (f[k]-f_ave)*(f[k]-f_ave);
		r_bunbo_1 += (t[k]-t_ave)*(t[k]-t_ave);
	}

	R=r_bunsi / ( sqrt(r_bunbo) * sqrt(r_bunbo_1) );


	return R;

}
```

[くし型フィルタ]

```
double comb(double x,int i,int p)
{
			double y = (1-a)*x + a*yout[i][p][  ( time[i][p]-N[i][p]+buf_length ) % buf_length ];
			yout[i][p][time[i][p]]=y;
			time[i][p]=(time[i][p]+1)%buf_length;

			return y;
}
```

[自己相関]

```
double phi(double* x, int N, int n)
{
        int i;
        double R;

        R=0.0;
        for(i=0;i<N-1;i++)
        {
                if( ( i+n ) < N )   R=R+x[i]*x[(i+n)];        //←/循環させない場合
                  （ R=R+x[i]*x[(i+n)%N]; ）                  //←循環する場合
        }
        R=R/N;

        return R;
}
```