平成23年度 博士論文

図形と音声の変換手法とその応用に関する研究

岩淵 勇樹
金沢大学大学院 自然科学研究科 電子情報科学専攻
集積回路工学研究室
学籍番号 0923112101
主任指導教員名: 秋田 純一教授

目 次

第1章　序論

1.1　はじめに

電子楽器の発明以降，それまでの物理的な振動現象を用いてきた楽器の音色は格段に生成の自由度が高くなり，今までにない新たな音色を作ることが可能となった．それに伴ってテクノ音楽等の新たな音楽分野が生まれ，それが再び新たな電子楽器の発展を促している．

電子楽器に向けた，楽器を演奏するための入力インタフェースに関する研究は多くあるが，それと並んで，電子楽器に適した音色の入力・生成方法，特にマウス操作やタッチパッド操作など GUI に適した音色入力・生成方法を探ることは，電子音楽全体の発展のためにも重要な課題といえる．

電子楽器の特性を生かした音色の生成方法は数多く研究されており，また実用化されているものも多いが，それらの多くは正弦波や矩形波などの単純な波形を組み合わせる加算合成方式や FM 音源方式のように，多数のパラメータを必要とする．そのため，意図した音色を生成するためには多くの経験を要する．また正弦波や三角波などの波形や PCM 録音されたサンプリング音源に対して，演算やフィルタリングによって音色を生成する減算合成方式も，同様にパラメータ制御が必要である．またこれらのパラメータは一次元量であり，その操作はスライダやツマミ等のインタフェースによって操作されるが，これらのインタフェースは，マウスやタッチパッドなどの近年一般的となっている平面入力・操作デバイスを有効に活用できているとはいえない．

我々はこれまで，マウスやタッチパッド等の平面入力・操作デバイスによる音色入力インタフェースを構築するために，シルエット画像などの図形を，解析信号と呼ばれる信号に変換して周期信号を生成する手法を提案してきた [1][2]．この方式では，平面入力デバイスによって描画した閉曲線図形を解析信号によって近似し，それに音色を生成するものであるが，解析信号に相当する閉曲線には幾何学的制約があるので，任意の閉曲線図形を解析信号で精度よく近似するには限界があり，ユーザインタフェースにより解決する必要がある．

以上をふまえ，本論文では，既存の音声信号をベースとし，それに対してマウスやタッチパッドなどの平面操作デバイスによって波形を操作することで，新たな音色の音声信号を作成する手法の提案と実装を行う．具体的には，既存の音声信号から生成された解析信号を複素平面上の閉曲線図形とみなし，これに対して平面操作デバイスによる図形操作によって新たな周期音声信号を生成・変更する手法の提案と実装を行う．また，本手法に地表面軌道合成 [3][4] および Scanned Synthesis[5][6] の方式を取り入れた拡張を行い，音声信号をもとに地表面軌道合成を施すことで，さらなる音のバリエーションを生成する手法について述べる．

1.2　背景

現在利用されている多くの音色操作インタフェースは，各種パラメータをツマミやスライダ等で操作するものであるが，これは 1 次元的なインタフェースであるといえる．

図 1.1: Kaossilator[7]

一方，計算機への入力デバイスとしては，従来からのマウスやトラックボールのほか，近年普及してきたタッチパッドなどがあるが，これらは 2 次元平面上でのポインタ移動や図形描画が可能である．

このような音色操作を実現したインタフェースの例として Kaossilator[7] が挙げられるが，これはピッチとエフェクトの強さの 2 つのパラメータを x 軸と y 軸に割り当てたものであり，2 つのパラメータを同時に操作可能であるものの，平面上の操作と生成される音色との関係を直感的に関連付けることは容易ではない．

平面入力インタフェースに適した音色の操作方法を探ることは，より豊かな音色生成のために有用であると考えられる．これらの入力デバイスを活用する音色操作インタフェースとして最適なのは，一定領域内の 2 次元平面上に配置された制御点を，操作するポインタによりクリックやドラッグなどの操作ができることであると考えられる．

1.3 関連研究

1.3.1 平面情報の時間-ピッチへの変換

1972 年に，イアニス・クセナキスによってタブレットボードに描いた線画図形の縦軸をピッチ，横軸を時間とみなして音声信号に変換する装置 UPIC が考案され，図形から音声信号への変換という分野が開拓された．Kaossilator[7] のように，平面タッチパッドをシンセサイザーの 2 つの独立した音色パラメータの入力に用いる装置も現れた．

音の可視化技術としてスペクトログラムがあるが，これも横軸が時間，縦軸がピッチ（周波数）に対応するという点では UPIC と同様である．実際，可視化の逆として，スペクトルグラムをラス

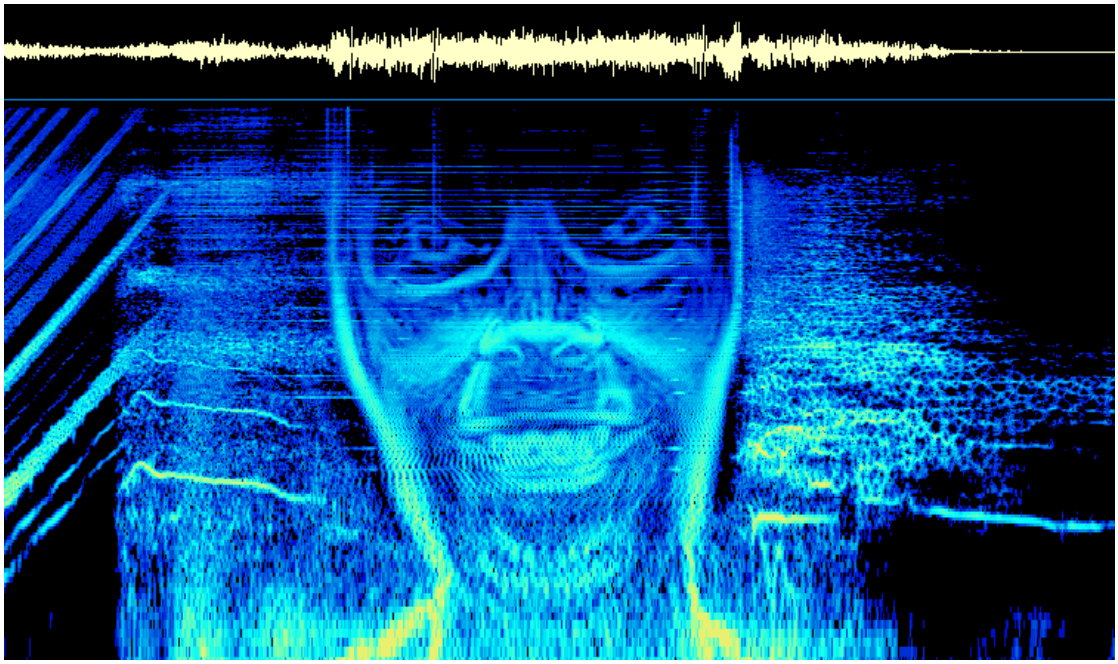


図 1.2: 画像から生成された音楽のスペクトログラムの例 [9]

タ画像として編集して音声信号を得る MetaSynth[8] などのソフトウェアが開発されている．エイフェックス・ツインなどの音楽家が実際にこれらのソフトウェアを用い、幾何学図形や写真に基づいた楽曲を発表している [9]（図 1.2）。ただし，スペクトログラムに画像情報を用いる場合，高周波成分も多く含まれることになり，金属的な音になりやすい．

また，動画投稿サイトでは 2007 年から The Music Animation Machine[10] などを用いた “MIDI アニメ” と呼ばれる作品が数多く投稿されており，UPIC と楽譜（ピアノロール）の中間的存在ともいえる．

TENORI-ON[11] も横軸を時間，縦軸をピッチとして作曲することができる．

1.3.2 平面情報を用いたその他の音声信号化

その他 GUI の特性を活かした例として，一部の効果音作成ソフトウェア等では波形をマウスで描画して音色を作る機能を備えているものもあり [12]，パラメータ入力よりは直感的に操作できるが，生の音声波形は視覚的に音色を想起させるに優れているとも一概には言い難い．また，この手法は信号の端点が不連続になったり波形が急峻になったりし易く，歪んだ音となり易いため，一般的な作曲にはあまり適さない．

ラスタ画像の輝度値を音声信号に変換する研究もある [13].

一方，メディアアートにおいては，図形を基にした音色生成の先行研究のひとつとして Levin らの作品 [14] が挙げられる．この作品では図形と音色の調和を意識して構成されているため直感的ではあるが，面積や周囲長などの画像の特徴量をパラメータ化しているに過ぎないため，形状情報が音色に寄与する割合は低く，表現力にも限界がある．

図 1.3: TENORI-ON[11]

1.3.3　ジェスチャ入力

マウスやタッチパッドなどの平面操作は，2 次元空間の入力と捉えることもできる．そもそも，電子音楽の祖であるテルミンも一種の空間情報を用いた楽器インタフェースであり，更に遡れば管弦楽器などもピッチを（3 次元空間上での 1 次元の）空間情報を用いて制御している．空間情報を用いた音色生成という観点からすると，任意の画像や図形から音色を生成するものよりは，人間のジェスチャを解析して入力インタフェースとして用いるものの方が多い．

楽器の模擬としてのインタフェースも数多く研究されているが，ジェスチャ入力を用いる BioMuse[15]，Very Nervous System[16]，EyesWeb[17] やユーザの表情を用いる SoFA[18]，口の形を用いる Mouthesizer[19][20]，などのように，体の動きそのものを楽器インタフェースとして用いるものも増えている．

1.4　本論文の構成

第 1 章 序論

音色入力インタフェースに関する背景および関連研究について述べる．

第 2 章 回転によって音色が不変である信号

音声を閉曲線図形として扱うとき，回転してもその実部の音声信号の音色が不変である信号について述べる．この条件を満たす信号として解析信号を挙げ，解析信号と関わりの深いヒルベルト変換について述べる．

第 3 章 音声信号から図形への変換

解析信号を用いて音声信号を閉曲線図形に変換する方法について述べる．また，解析信号となるような閉曲線について図形的特徴量を求め，どのような違いがあるかを示す．

第 4 章 図形から音声信号への変換

閉曲線図形を解析信号となるような周期音声信号に変換する方法について述べる．

第 5 章 解析信号シンセサイザ“ CloSynth ”の開発

解析信号を用いた音色の操作インタフェースについて述べる．具体的には，音色を平面上の閉曲線として取り扱い，その図形に対する変形等の操作によって音色の操作を行う手法の提案と実装を行う．また，CloSynth の拡張として Scanned Synthesis の手法を取り入れる．

第 6 章 まとめ

本研究の総括をすると共に，展望について述べる．

第2章　回転によって音色が不変である信号

2.1　解析信号

$U(\omega)$ を以下のように定義する．

$$U(\omega) = \begin{cases} 0 & (\omega < 0) \\ 1 & (\omega = 0) \\ 2 & (\omega > 0) \end{cases} \tag{2.1}$$

音声信号 $s(t)$ に対して，そのフーリエ変換を $S(\omega)$ とおく．このとき，$U(\omega)$ をフィルタとして用いると，式 (2.2) のように，$S(\omega)$ の正周波数成分の振幅が 2 倍，負周波数成分の振幅が 0 倍となり，それを $\tilde{S}(\omega)$ とおく．

$$\tilde{S}(\omega) = U(\omega) \cdot S(\omega) = \begin{cases} 0 & (\omega < 0) \\ S(\omega) & (\omega = 0) \\ 2S(\omega) & (\omega > 0) \end{cases} \tag{2.2}$$

それを逆フーリエ変換した複素信号を一般に「解析信号」と呼び，$\tilde{s}(t)$ と表す．

$s(t)$ が区分的に滑らかで常に有限値をもつ周期信号であれば，$\tilde{s}(t)$ は複素平面 $x + iy$ 上の閉曲線を描くことが導かれる．(ただし i は虚数単位)．図 2.2 に解析信号の例を示す．

2.2　解析信号と音色の関係

周期的な解析信号を複素平面上の閉曲線図形として見ると，音声信号の振幅が時間とともに変化するのにあわせて，それに対応する解析信号の閉曲線上を対応点が移動し，音声信号の 1 周期が閉曲線上の 1 周に対応する．すなわちこの閉曲線の形状は，音声信号の周波数情報を除いた表現であると言うことができる．

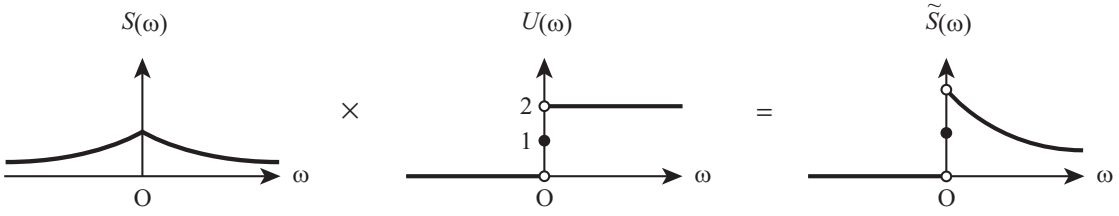


図 2.1: 解析信号フィルタを掛け合わせた信号の概念図

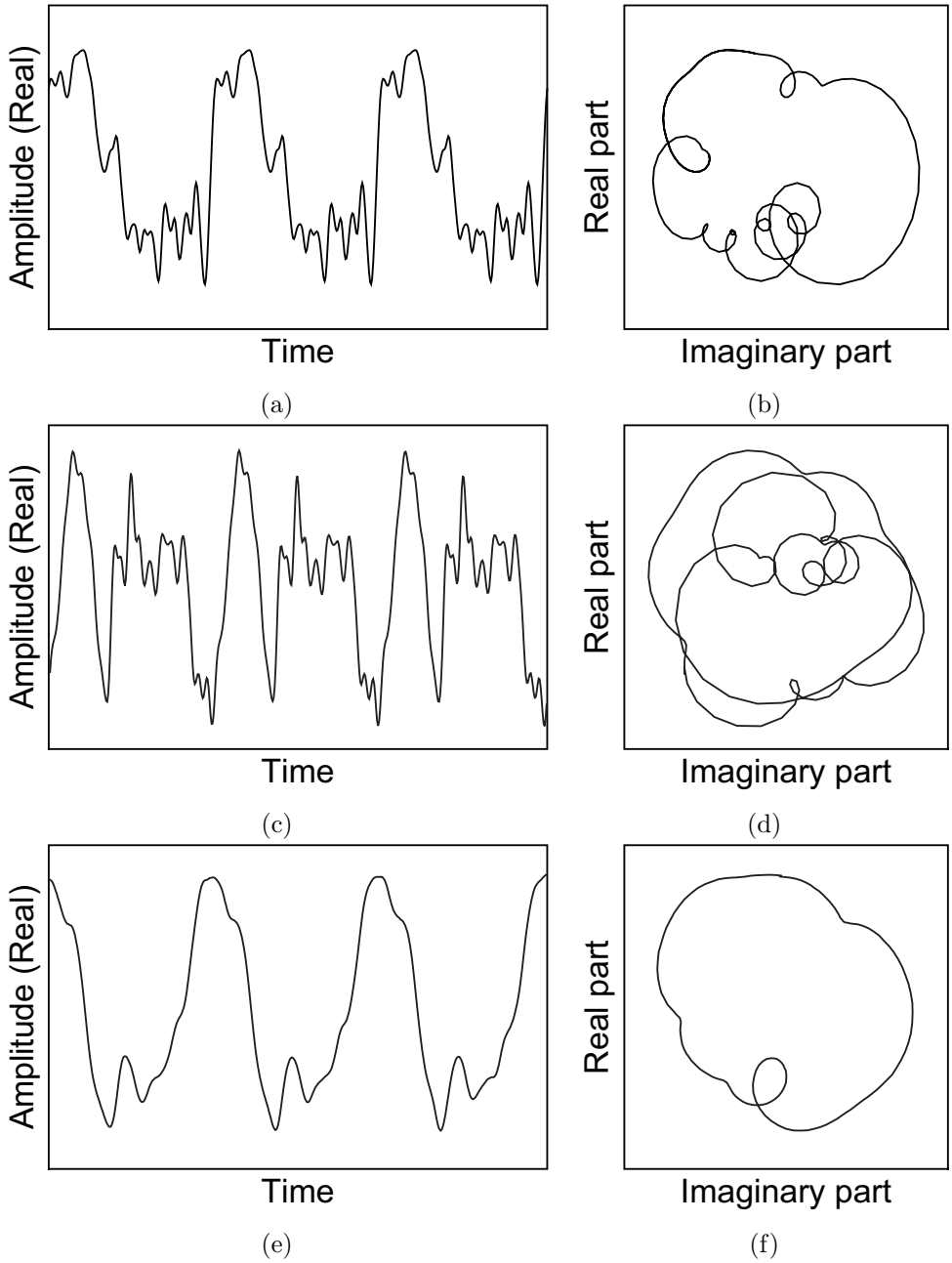


図 2.2: バイオリンの音色 $s(t)$(a) と対応する解析信号 $\tilde{s}(t)$(b)，ハーモニカの音色 $s(t)$(c) と対応する解析信号 $\tilde{s}(t)$(d)，ピアノの音色 $s(t)$(e) と対応する解析信号 $\tilde{s}(t)$(f)

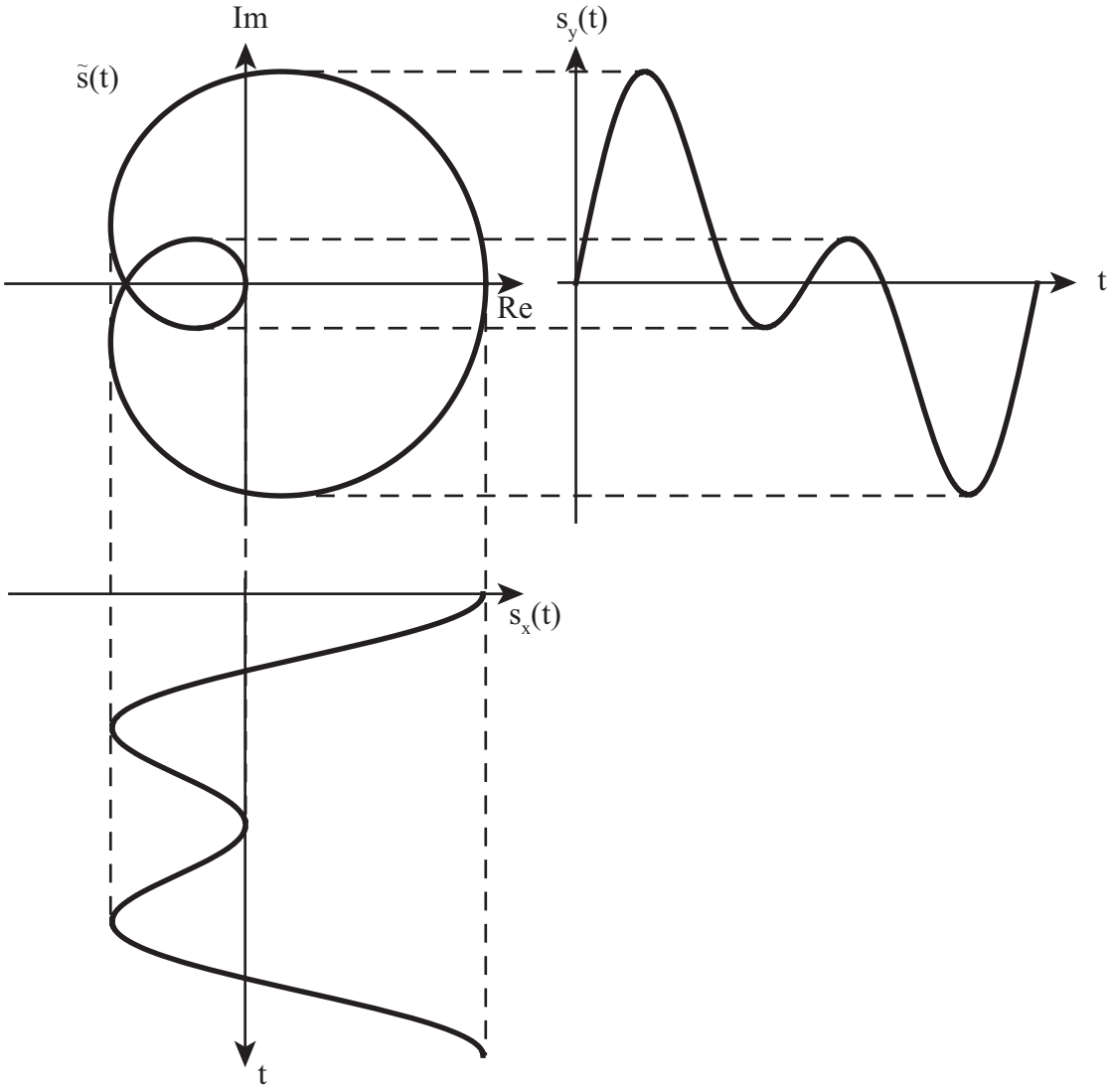


図 2.3: 解析信号とその実部および虚部

閉曲線の大きさが音声信号の振幅に対応し，閉曲線の平行移動は，時間変化しない定数成分に対応するため音色は変化しない．また位相回転 (定数 $e^{i\theta}$ 倍) は閉曲線の回転に対応するが，一般に音色は基音の位相に依存しない．そのため，相似な閉曲線はすべて同じ音色に対応することになる．

以上のことから，閉曲線図形としての解析信号は音色と密接な関わりをもっているといえる．

2.3 ヒルベルト変換

$F(\omega)$ を以下のように定める．

$$F(\omega) = \begin{cases} i & (\omega < 0) \\ 0 & (\omega = 0) \\ -i & (\omega > 0) \end{cases} \tag{2.3}$$

このとき，

$$H(\omega) = F(\omega) \cdot S(\omega) \tag{2.4}$$

とすると，$H(\omega)$ の逆フーリエ変換 $h(t)$ は $s(t)$ のヒルベルト変換であるという．$\tilde{s}(t)$ の実部 $\Re[\tilde{s}(t)]$ は $s(t)$ のままであるという性質をもち，$h(t)$ は解析信号 $\tilde{s}(t)$ の虚部 $\Im[\tilde{s}(t)]$ に等しくなる．

$$\Re[\tilde{s}(t)] = s(t) \tag{2.5}$$

$$\Im[\tilde{s}(t)] = h(t) \tag{2.6}$$

なお，$s(t)$ は $h(t)$ の逆ヒルベルト変換であるといい，$F(\omega)$ の代わりに $-F(\omega)$ を用いた時の $s(t)$ と $h(t)$ の関係に等しい．式 (2.4) は畳み込みの関係を用いて以下のように表すことができる．

$$h(t) = \frac{1}{\pi t} * s(t) \tag{2.7}$$

また，$\tilde{s}(t)$ は負の周波数成分を持たないという性質があり，すなわち解析信号である．なお，$s_x(t)$ は $s_y(t)$ の逆ヒルベルト変換であるといい，$F(\omega)$ の代わりに $-F(\omega)$ を用いた時の $s_x(t)$ と $s_y(t)$ の関係に等しい．

2.4 回転によって音色が不変であるということ

$\tilde{s}(t)$ が解析信号であるとき，以下の等式が成り立つ（$\mathcal{F}$ はフーリエ変換を表す）．

$$\left|\mathcal{F}[\Re[e^{i\theta}\tilde{s}(t)]]\right| = |S_x(\omega)| \tag{2.8}$$

ところで，人間が周期的な音声信号を知覚する際，聞こえに影響するのはほとんど振幅スペクトルの形状の違いであり，位相が与える影響はごく僅かである．そのため，式 (2.8) より，周期関数 $s_x(t)$ および $\Re[e^{i\theta}\tilde{s}(t)]$ は同等の音色をもつとみなす．すなわち，複素平面上の閉曲線が与えられたとき，その実部を音声信号に対応させるとすると，閉曲線が解析信号であった場合はその回転図形も同等な音色をもつことになる．

また，音声信号の定常成分は音色に全く影響を与えないため，定常成分のみ異なる音声信号もそれぞれ同等な音色をもつと見なせる．なお，本論文では定常成分を含まない信号のみを考える．

以上を総合すると，本論文における音色は，定常成分を除く振幅スペクトルの形状によって特徴付けられる．

2.4.1 曲線図形の多様性

任意の単調増加写像 g によってパラメータ変換された曲線 $\tilde{s}(g(t))$ の軌跡は，$\tilde{s}(t)$ の軌跡と等しい．よって，閉曲線が与えられたとき，パラメータ変換によって解析信号を生成することが可能ならば，任意の閉曲線図形の入力に対して回転しても生成される音色が不変な音声信号生成が可能といえる．さらに，図形の大きさは音声信号の振幅，つまり音量に相当するため，相似図形については同一の音色が対応することになり，形状特徴に固有な変換といえる．

しかしながら，筆者の予想では，解析信号が閉曲線図形として表現できる範囲には限界があり，そのようなパラメータ変換がほとんどの場合において不可能だと考えられる．

そのため，式 (2.8) が恒等的に成立するための条件を求め，閉曲線の表現範囲を拡張する．

2.4.2 回転によって音色が不変である条件

任意の複素数 w について

$$\Re[w] = \frac{w + \overline{w}}{2} \tag{2.9}$$

が成立することを用いると，$\Re[e^{i\theta}\tilde{s}(t)]$ は以下のように変形できる．

$$\begin{aligned}
& \Re[e^{i\theta}\tilde{s}(t)] = \frac{e^{i\theta}\tilde{s}(t) + \overline{e^{i\theta}\tilde{s}(t)}}{2} \\
= \quad & \frac{e^{i\theta}\{s_x(t) + is_y(t)\} + e^{-i\theta}\{s_x(t) - is_y(t)\}}{2} \\
= \quad & \frac{(e^{i\theta} + e^{-i\theta})s_x(t) + i(e^{i\theta} - e^{-i\theta})s_y(t)}{2} \\
= \quad & \cos\theta \cdot s_x(t) - \sin\theta \cdot s_y(t)
\end{aligned} \tag{2.10}$$

そして，これのフーリエ変換の絶対値を 2 乗したものは以下のように変形できる．

$$\begin{aligned}
& |\cos\theta \cdot S_x(\omega) - \sin\theta \cdot S_y(\omega)|^2 \\
= \quad & \{\cos\theta \cdot S_x(\omega) - \sin\theta \cdot S_y(\omega)\} \left\{\cos\theta \cdot \overline{S_x(\omega)} - \sin\theta \cdot \overline{S_y(\omega)}\right\} \\
= \quad & \cos^2\theta\,|S_x(\omega)|^2 + \sin^2\theta\,|S_y(\omega)|^2 - \cos\theta\sin\theta \left\{S_x(\omega)\overline{S_y(\omega)} + \overline{S_x(\omega)}S_y(\omega)\right\} \\
= \quad & \cos^2\theta\,|S_x(\omega)|^2 + \sin^2\theta\,|F(\omega)S_x(\omega)|^2 - \cos\theta\sin\theta \left\{S_x(\omega)\overline{F(\omega)S_x(\omega)} + \overline{S_x(\omega)}F(\omega)S_x(\omega)\right\} \\
= \quad & |S_x(\omega)|^2 \left\{\cos^2\theta + \sin^2\theta\,|F(\omega)| + \cos\theta\sin\theta\,\{\Re[F(\omega)]\}\right\}
\end{aligned}$$

そして，上式は恒等的に $|S_x(\omega)|^2$ になる必要があるため，以下の条件を満たさなければならない．

$$\begin{cases} |F(\omega)| = 1 \\ \Re[F(\omega)] = 0 \end{cases} \tag{2.11}$$

これはつまり絶対値が 1 である純虚数，$\{i, -i\}$ のみを値域としてもつ関数である．また，$s_x(t)$ と $s_y(t)$ はともに実関数であるため，$F(\omega)$ は奇関数である必要がある．

この関係より，任意の $\omega' > 0$ について $S_x(\omega') + iS_y(\omega')$ か $S_x(-\omega') + iS_y(-\omega')$ のどちらかは打ち消しあって 0 になるため，最終的に $\tilde{s}(t)$ のスペクトルは「周波数毎に正か負どちらかのみの周波数成分をもつ」ものとなる．

また，周期関数であるため，複素係数列 a_n と符号列 $s_n(\in \{-1, 1\})$ を用いて以下のような級数として表現することができる．

$$\sum_{n=1}^{\infty} a_n e^{(s_n i n \omega_0 t)} \tag{2.12}$$

第3章　音声信号から図形への変換

3.1　音声信号と曲線

音声信号を $s_x(t)$ とし，そのフーリエ変換を $S_x(\omega)$ とする．このとき，$S_x(\omega)$ にあるフィルタ $F(\omega)$ を掛けて

$$S_y(\omega) = F(\omega) \cdot S(\omega) \tag{3.1}$$

となる $S_y(\omega)$ を与えると，その逆フーリエ変換 $s_y(t)$ と原信号 $s_x(t)$ を用いて

$$\tilde{s}(t) = s_x(t) + is_y(t) \tag{3.2}$$

という複素関数が定まる（i は虚数単位）．$F(\omega)$ が定数の場合などを除けば，$\tilde{s}(t)$ は複素平面 $s_x + is_y$ 上の曲線となり，特に $s_x(t)$ が周期関数かつ $s_x(t)$ および $s_y(t)$ が区分的に滑らかで常に有限値の場合は閉曲線となる．

本論文では，フィルタ $F(\omega)$ を式 (2.3) のヒルベルト変換フィルタに限定して議論する．

3.2　周期音声信号から閉曲線への変換

周期信号を閉曲線に変換するには 2.1 で述べたように、信号を解析信号に変換して複素平面上に投影することによって描かれる．

3.3　音声信号と閉曲線の特徴量の関係

3.3.1　閉曲線の特徴量

解析信号であるような閉曲線 $\tilde{s}(t)(0 \leq t \leq 2\pi)$ の例として，閉曲線 $\tilde{s_1}(t), \tilde{s_2}(t)$ を以下のように定める．

$$\tilde{s_1}(t) = \sum_{n=1}^{10} e^{iA_n t - p_n} \tag{3.3}$$

$$\tilde{s_2}(t) = \sum_{n=1}^{10} e^{iB_n t - q_n} \tag{3.4}$$

$A_n = [1,1,1,1,1,0,0,0,0,0], B_n = [0,0,0,0,0,1,1,1,1,1]$ とし，p_n, q_n は 0 以上 2π 未満の実数とする．

$\tilde{s_1}(t), \tilde{s_2}(t)$ について、p_n, q_n をランダムに変化させることにより，以下の特徴量を各 100 回求める．

回転数　$\tilde{s}(t)$ の偏角（$-\pi$〜π）が π から $-\pi$ に変化する回数から，$-\pi$ から π に変化する回数を引いた数を回転数と定義する．

偏角極大値数　$\tilde{s}(t)$ の偏角（$-\pi$〜π）の極大値数．これは極小値数と一致する．

絶対値極大値数　$|\tilde{s}(t)|$ の極大値数．これは極小値数と一致する．

実部極大値数　$\Re[\tilde{s}(t)]$ の極大値数．これは極小値数と一致する．

1 周の長さ　曲線の 1 周分の長さ．

交点数　曲線の 1 点の座標がが他の 1 点の座標と一致する個数．

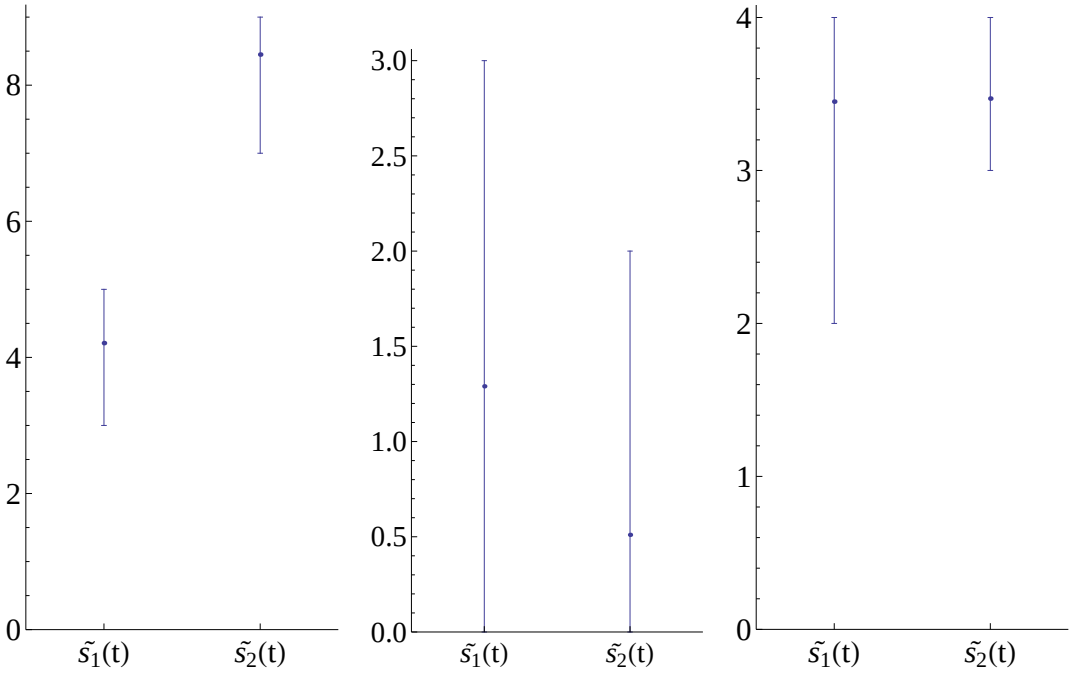
3.3.2　シミュレーション結果

得られた各特徴量を図 3.1 に示す．縦棒内の点が 100 回の平均値，上端が最大値，下端が最小値である．

偏角極大値数や絶対値最大値数ではあまり大きな差は得られなかったが，回転数，実部極大値数，1 周の長さ，交点数には大きな違いが表れた．

3.4　まとめ

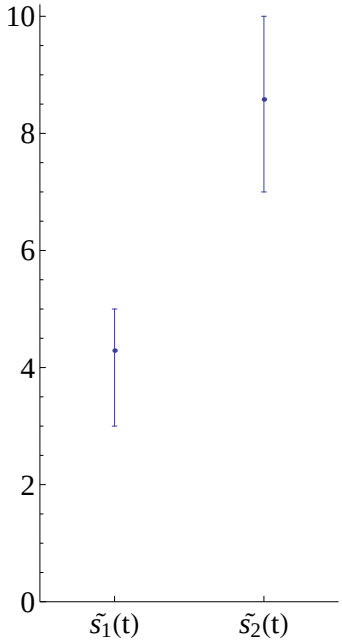
本章では音声信号を曲線図形へ変換する方法について述べ，2 種類の周期音声信号についてそれらを閉曲線に変換したときの特徴量の違いを示した．これらの特徴量を求めることにより，音色を自動分類するなどの応用が考えられる．



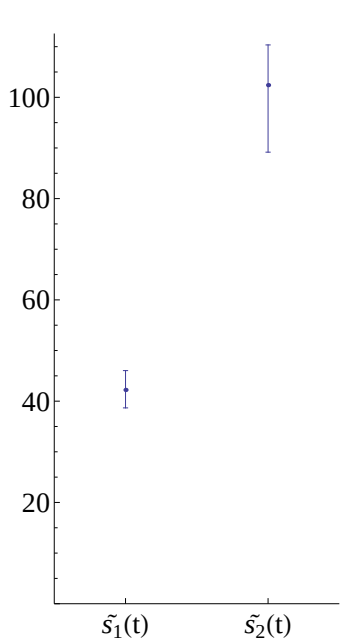
(a) 回転数

(b) 偏角極大値数

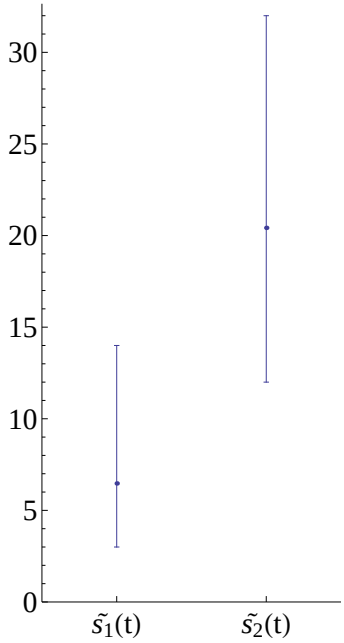
(c) 絶対値極大値数



(d) 実部極大値数



(e)1 周の長さ



(f) 交点数

図 3.1: 低周波と高周波の特徴量

第4章　図形から音声信号への変換

2.4.2 節にて曲線がもつ条件が拡張されたが，なおも任意の閉曲線図形を表現できるかどうかは自明でない．いずれにせよ，実装の際には式 (2.12) のような無限和を扱えないため，近似解を求める必要がある．

4.1　解析信号への近似

閉曲線を解析信号に漸近させる方法について検討する．

シミュレーションの初期曲線には，約 15 ピクセル角のシルエット画像を 8-連結の境界追跡によって求めた輪郭線を用いる．初期曲線は $m+1$ 個の複素平面上の点列 $\tilde{s}_0(t) = (\tilde{s}_0(0), \tilde{s}_0(1), \ldots, \tilde{s}_0(m))$ によって構成され，$\tilde{s}_0(m) = \tilde{s}_0(0)$ とする．フーリエ変換時にはこの重複する末尾の点を無視して FFT を行う．

初期曲線には図 4.1 に示す曲線 1 と曲線 2 を用いる．それぞれのスペクトルは (c)(d) のようになる．

4.1.1　方法 A: 解析信号対の漸近による方法

解析信号 $\tilde{s}(t)$ では，実部解析信号 $\xi(t) = s_x(t) + i\mathcal{H}[s_x(t)]$ と虚部解析信号 $\eta(t) = \mathcal{H}^{-1}[s_y(t)] + is_y(t)$ は当然等しくなる．なお，$\mathcal{H}$，$\mathcal{H}^{-1}$ はそれぞれヒルベルト変換，逆ヒルベルト変換を表す．

以下の手順で $\xi_{tmp}(t) - \eta_{tmp}(t)$ の値を最小化させ，入力曲線を解析信号に漸近させることを試みる．

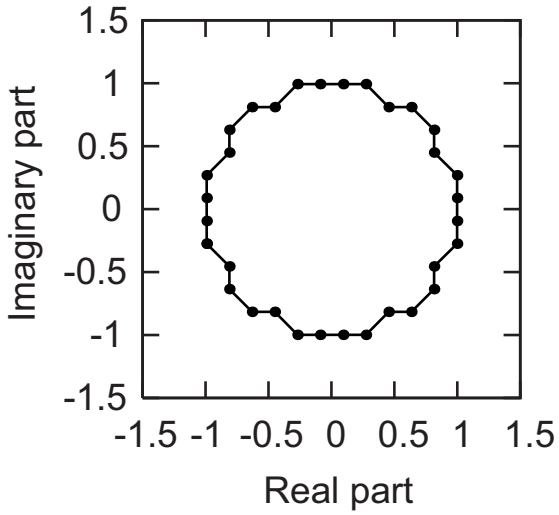
1. $n = 0$ とおく
2. $\tilde{s}_n(k)$ と $\tilde{s}_n(k+1)$ の間の位置に $\frac{\tilde{s}_n(k)+\tilde{s}_n(k+1)}{2}$ を挿入し，$\tilde{s}_{tmp}(t)$ とおく
3. $\tilde{s}_{tmp}(t)$ について，$\xi_{tmp}(t) - \eta_{tmp}(t)$ の 2 乗和が最小となる k を求め，そのときの $\tilde{s}_{tmp}(t)$ を $\tilde{s}_{n+1}(t)$ とする
4. n を $n+1$ に置き換え，2. 以降を繰り返す

この方法での $n = 100$ におけるシミュレーション結果を図 4.2, 4.3 に示す．

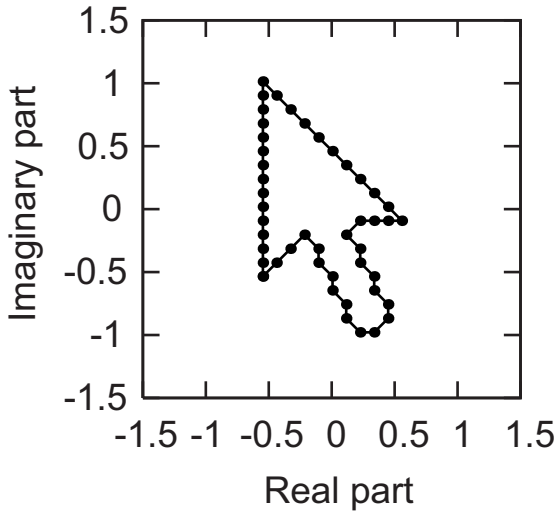
4.1.2　方法 B: 負周波数成分除去による方法

既に述べた通り，解析信号はその周波数成分のうち負周波数成分は全て 0 である．また，定数成分も含まないと考える．そのため，上記の方法のうち 3. における評価値を $\tilde{s}_{tmp}(t)$ の 0 以下の周波数成分の 2 乗和に置き換える．

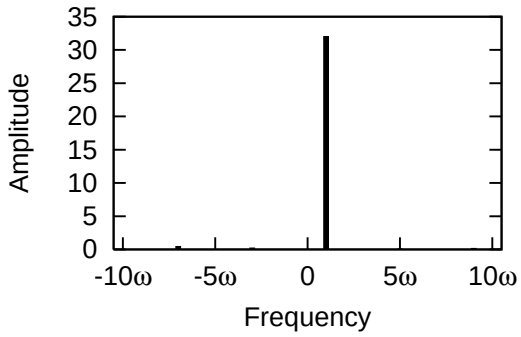
この方法での $n = 100$ におけるシミュレーション結果を図 4.4, 4.5 に示す．



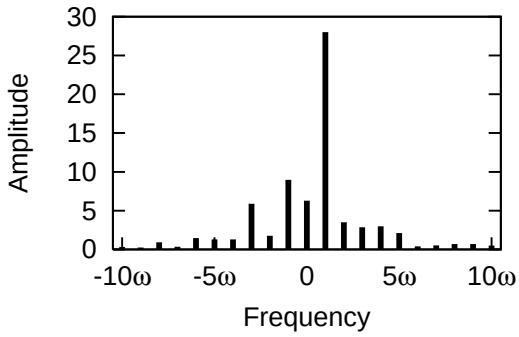
(a) 曲線 1



(b) 曲線 2

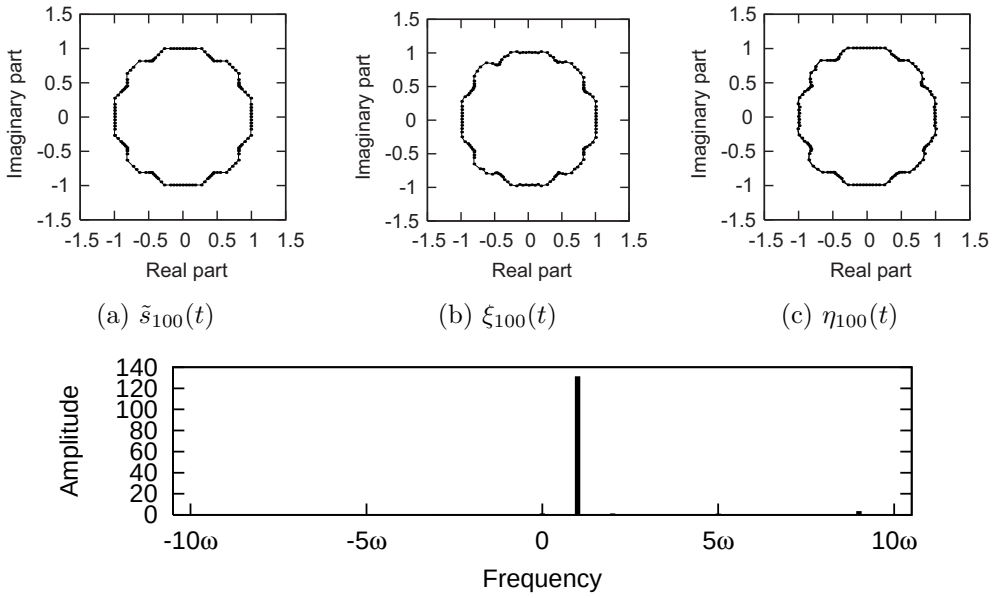


(c) 曲線 1 のスペクトル



(d) 曲線 2 のスペクトル

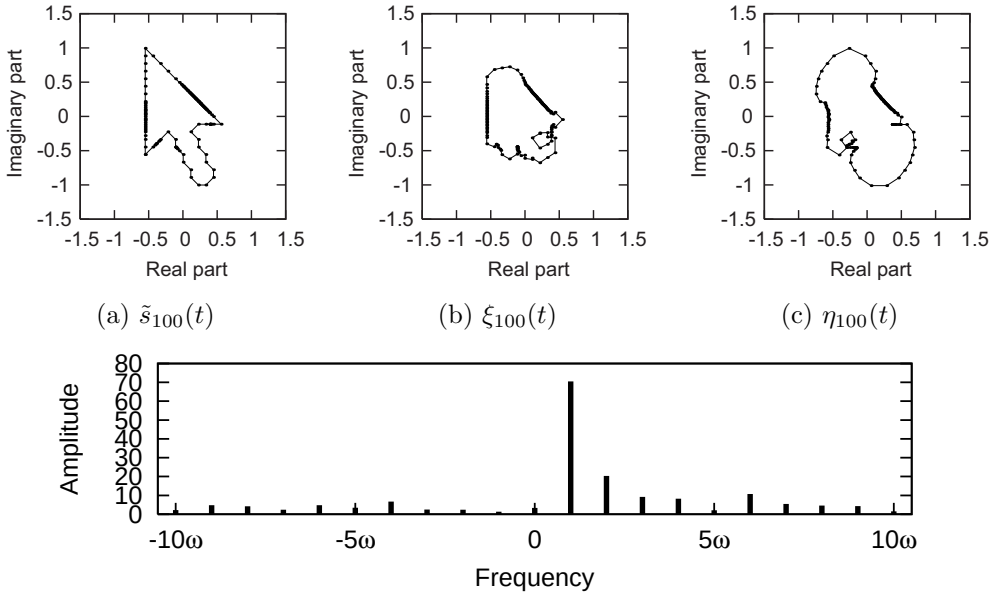
図 4.1: 入力図形とそのスペクトル



(a) $\tilde{s}_{100}(t)$ (b) $\xi_{100}(t)$ (c) $\eta_{100}(t)$

(d) $\tilde{s}_{100}(t)$ のスペクトル

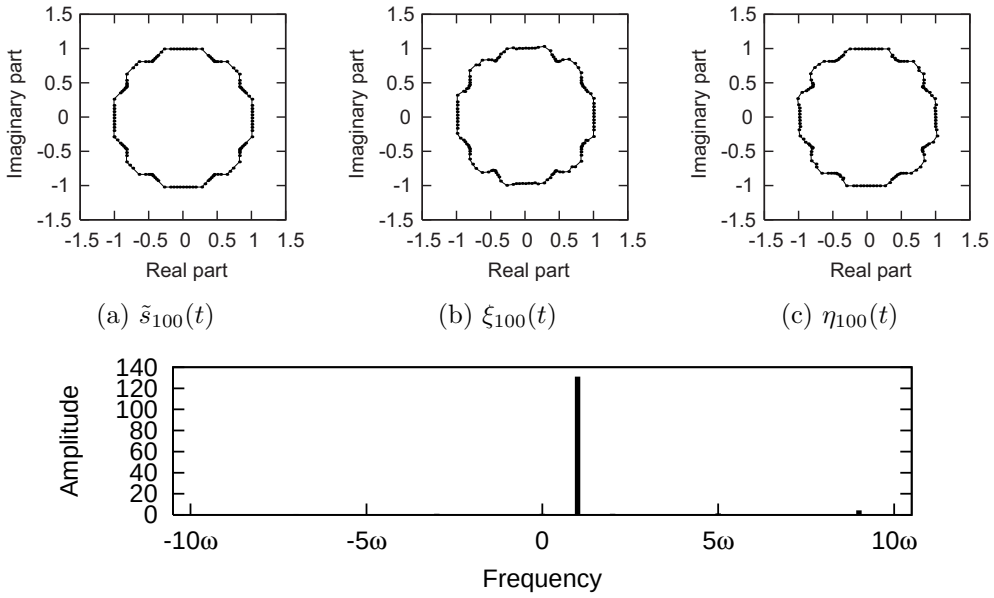
図 4.2: 方法 A のシミュレーション結果（曲線 1）



(a) $\tilde{s}_{100}(t)$ (b) $\xi_{100}(t)$ (c) $\eta_{100}(t)$

(d) $\tilde{s}_{100}(t)$ のスペクトル

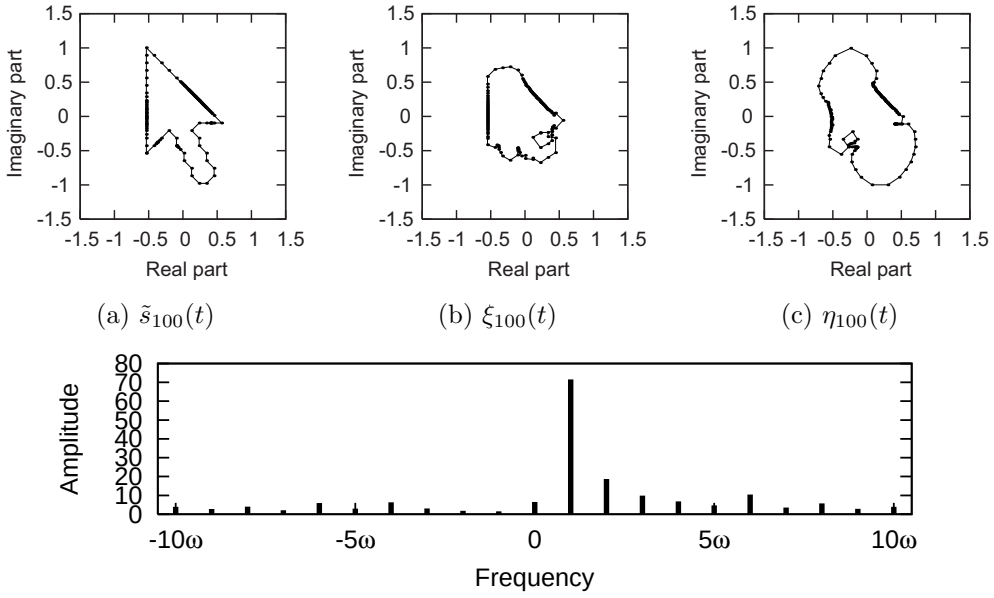
図 4.3: 方法 A のシミュレーション結果（曲線 2）



(a) $\tilde{s}_{100}(t)$ (b) $\xi_{100}(t)$ (c) $\eta_{100}(t)$

(d) $\tilde{s}_{100}(t)$ のスペクトル

図 4.4: 方法 B のシミュレーション結果（曲線 1）



(a) $\tilde{s}_{100}(t)$ (b) $\xi_{100}(t)$ (c) $\eta_{100}(t)$

(d) $\tilde{s}_{100}(t)$ のスペクトル

図 4.5: 方法 B のシミュレーション結果（曲線 2）

4.1.3 考察

以上，2 種類の図形の解析信号近似を試みた結果，2 つの方法でほとんど同じ結果が得られた．

曲線 1 については解析信号にかなり近い形となったが，これは元々入力曲線が解析信号に近かったことが大きな要因であると考えられる．曲線 2 については，0 以下のスペクトル成分をある程度減らすことは出来たが，$\xi_{100}(t)$ および $\eta_{100}(t)$ を入力曲線と見比べると，形状の見た目としては少しは似ているものの，特徴が大きく失われている．

今回のように曲線の点列に 1 点ずつ標本点を追加する場合，挿入点は総当たりによって決定されるため，点が増えるとともに計算量も増加することになり，効率も良くない．

また，挿入点に隣り合う値の平均値を挿入値としているため，これは入力点列に対する線形補間と捉えることもできるが，線形補間は理想的な補間方法とはいえず，滑らかにならずに結果として高周波成分を多く含むことになる．自然な音色は高周波成分を多く含まないため，これは望ましくない．今回用いたシルエット画像のように解像度の低い画像を用いる場合は，補間する点にある程度の自由度があるとなおよいと考えられる．

今回の方式を改良するならば，点の追加と削除を交互に行うなどして，標本点の個数を固定した上で最適化を行うことができればより理想的な結果が得られるのではないかと考えられる．

4.2 合成関数によるパラメータ変換

4.1 節では，サンプル点の平均点を追加する方法によってパラメータを変換し，近似を試みた．この方法では挿入時に線形補間が行われるため，理想的な近似解析信号とのずれが生じるほか，内挿数が増えるにしたがって計算量が増加するという問題がある．このため，サンプル点を維持して合成関数を変換することによる近似を考える．

4.2.1 合成関数のフーリエ級数展開

$$\mathcal{F}(f(t)) := \int_0^1 f(t)e^{-2\pi int}\,dt \tag{4.1}$$

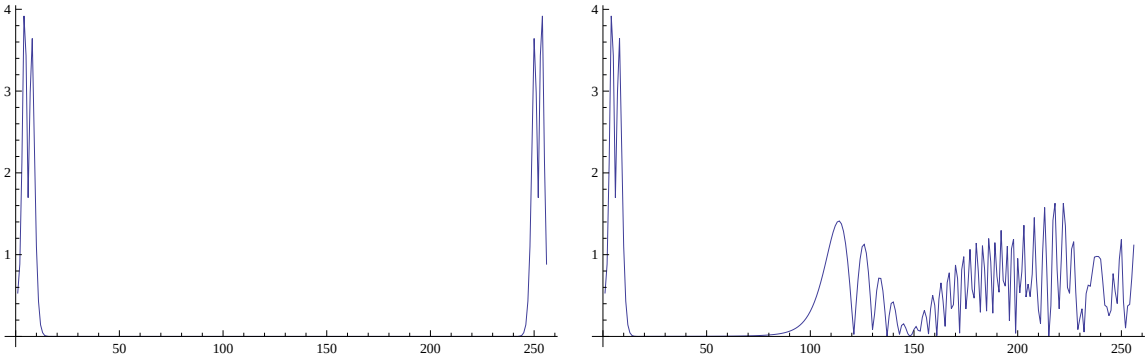
ここで $x := g(t)$ とすると，合成関数 g によってパラメータ変換を行った関数のフーリエ変換は以下のようになる．

$$\begin{aligned}\mathcal{F}(f(g(t))) &= \int_0^1 f(g(t))e^{-2\pi int}\,dt &(4.2)\\ &= \int_0^1 f(x)e^{-2\pi ing^{(-1)}(x)}g^{(-1)'}(x)\,dx &(4.3)\end{aligned}$$

4.2.2 合成関数の離散フーリエ変換

前節の定義を離散フーリエ変換に置き換えると，以下のように表すことができる．

$$\mathcal{F}(f(t)) := \frac{1}{\sqrt{T}}\sum_{t=0}^{T-1} f(t)e^{2\pi in\frac{t}{T}} \tag{4.4}$$



(a) 合成後に離散フーリエ変換　　(b) 提案法

図 4.6: 合成関数の離散フーリエ変換

これを合成関数 g によってパラメータ変換を行った関数のフーリエ変換は以下のようになる．

$$\mathcal{F}(f(g(t))) = \frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} f(g(t)) e^{2\pi i n \frac{t}{T}} \tag{4.5}$$

これをフーリエ級数展開と同様の変形を行うと以下のようになる．

$$\mathcal{F}(f(g(t))) = \frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} f(t) e^{2\pi i n g^{(-1)}\left(\frac{t}{T}\right)} g^{(-1)'}\left(\frac{t}{T}\right) \tag{4.6}$$

しかし，これは近似的な変形であり，正確な等式ではない．

実際この通りに変換した結果が図 4.6 である．低周波数域ではほぼ等しいが，高周波数域においてノイズが発生している．

4.3 まとめ

閉曲線を解析信号に変換する方法についてのシミュレーションを行った．

今後は式 (2.12) を満たす，解析信号を拡張した曲線を求める近似解の導出方法ついて検討したい．

第5章　解析信号シンセサイザ“ CloSynth ”の開発

5.1　周期解析信号に対するインタラクティブ操作

5.1.1　解析信号の離散化と操作の定義

まず音声信号 s を，s_0 から s_{n-1} までの n 個のサンプル点によって離散化する．s はこれら n 個のサンプル点列を周期とすると周期信号であると仮定し，$s_{k+mn} = s_k$ (m は任意の整数) とする．そして，これらの各サンプル点に対して解析信号 $\tilde{s}_0$〜$\tilde{s}_{n-1}$ を求めて複素平面上に表示し，各点を制御点として滑らかに結んだ閉曲線を生成する．この閉曲線は，元の音声信号の解析信号 $\tilde{s}$ の近似とみなすことができる．

このうちの 1 つの制御点 $\tilde{s}_k$ をマウス等でドラッグすることで移動させる際に，$\tilde{s}$ には $\tilde{s}_k$ を中心とした解析信号 $\tilde{p}$ を印加するとする．ここで $\tilde{p}$ も $\tilde{s}$ と同様に n 個のサンプル点から成る複素平面上の信号である．この制御点の移動先を $\tilde{s}_q$ とおき，新たに生成される $\tilde{s}'$ を以下のように定める．

$$\tilde{s}'_j = \tilde{s}_j + (\tilde{s}_q - \tilde{s}_j)\tilde{p}_{j-k} \tag{5.1}$$

なおこの制御点の移動中は，その移動に従って新たな音声信号 $s'(= \Re[\tilde{s}'])$ が適当な周波数の周期音声信号となるよう合成する．

ここで，$\tilde{p}$ を，その実部が以下のようになるように定める．

$$p_k = \left(\frac{1 + \cos\frac{2\pi k}{n}}{2} \right)^d \tag{5.2}$$

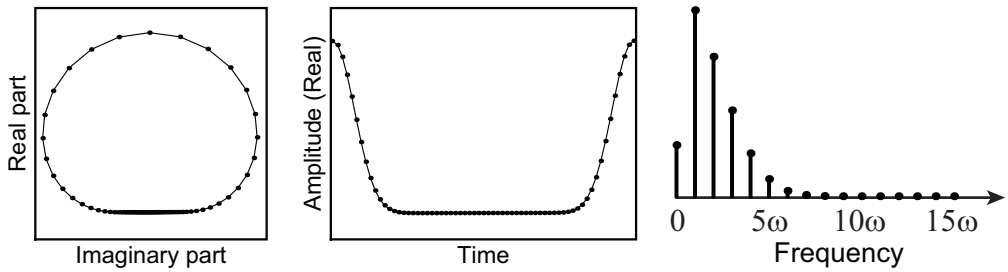
ここで d は 1 以上の実数で，制御点の移動の際に自由に設定できるとする．

$n = 64, d = 10$ の時の印加信号を図 5.1 に示す．実信号としては 0 に近い値が半数以上連続しており，解析信号としては円形が潰れたような形となる．

式 (5.2) の定義に従えば，$\tilde{p}_0 = 1$ になり，また同時に $\tilde{s}'_k = \tilde{s}_q$ となるため，移動中の制御点は常に操作指示点 (マウスによるドラッグ操作時のマウスポインタ等) と同位置になる．なお解析信号は線型性をもつため，印加解析信号を加えた後の信号もまた解析信号となる．

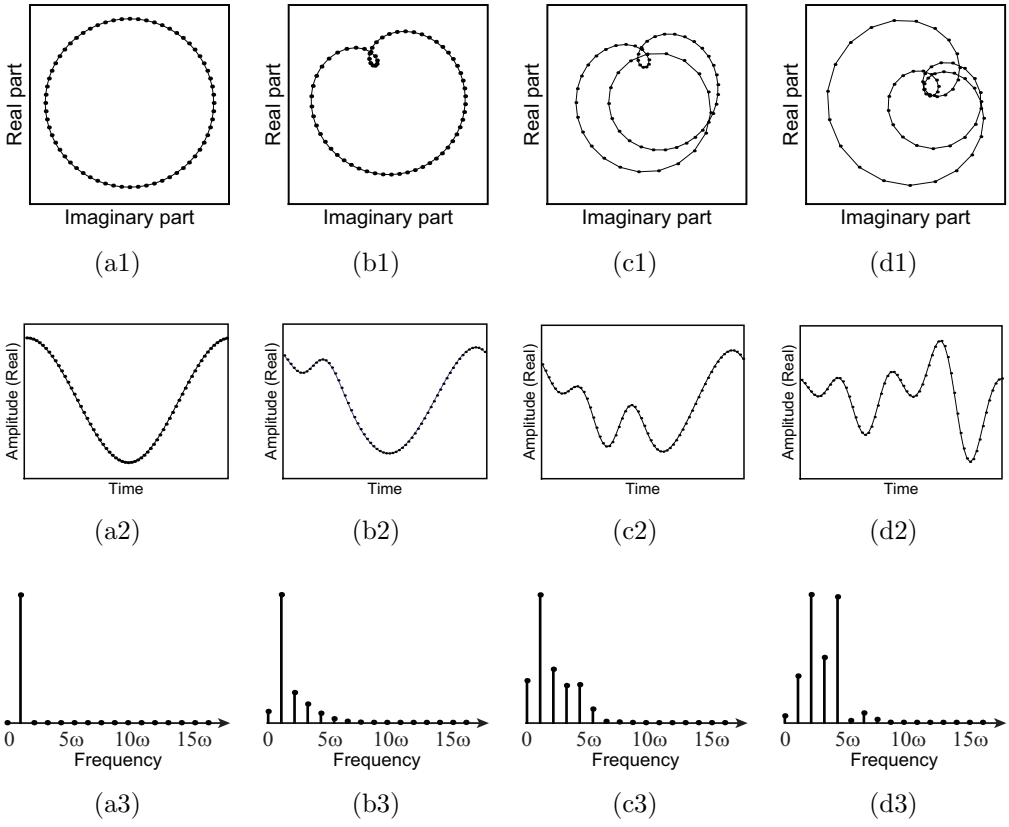
音色操作の例として，正弦波波形に対応する解析信号 (円) を最初に用意し，それにに対して，$n = 64, d = 10$ として 3 回の制御点移動操作を施したときの解析信号の変化，およびそれらに対応する 1 周期分の時間軸上の音声信号を図 5.2 に示す．

本手法は，制御点の操作は特定の信号の加算に対応するため，いわゆる加算合成方式のシンセサイザの一種ともいえる．基本的には，制御点を閉曲線の内側に「押し込める」操作によって基本周波数成分が減少・高周波数成分が増加し，曲線にねじれが生じる．また制御点を閉曲線の外側に「拡げる」操作によって基本周波数成分が増加・高周波数成分が減少し，曲線のねじれが解消される．



(a) (b) (c)

図 5.1: $n = 64, d = 10$ の時の印加信号の解析信号 (a) と，それに対応する音声信号 (b) とそのスペクトル (c)



(a1) (b1) (c1) (d1)

(a2) (b2) (c2) (d2)

(a3) (b3) (c3) (d3)

図 5.2: 正弦波の解析信号 (a1)，それに対応する時間軸波形の音声信号 (a2) とそのスペクトル (a3)，およびこれに対して $n = 64, d = 10$ としてドラッグ操作を順に施したときの解析信号 (b1)〜(d1)，それらに対応する時間軸波形の音声信号 (b2)〜(d2)，およびそれらのスペクトル (b3)〜(d3).

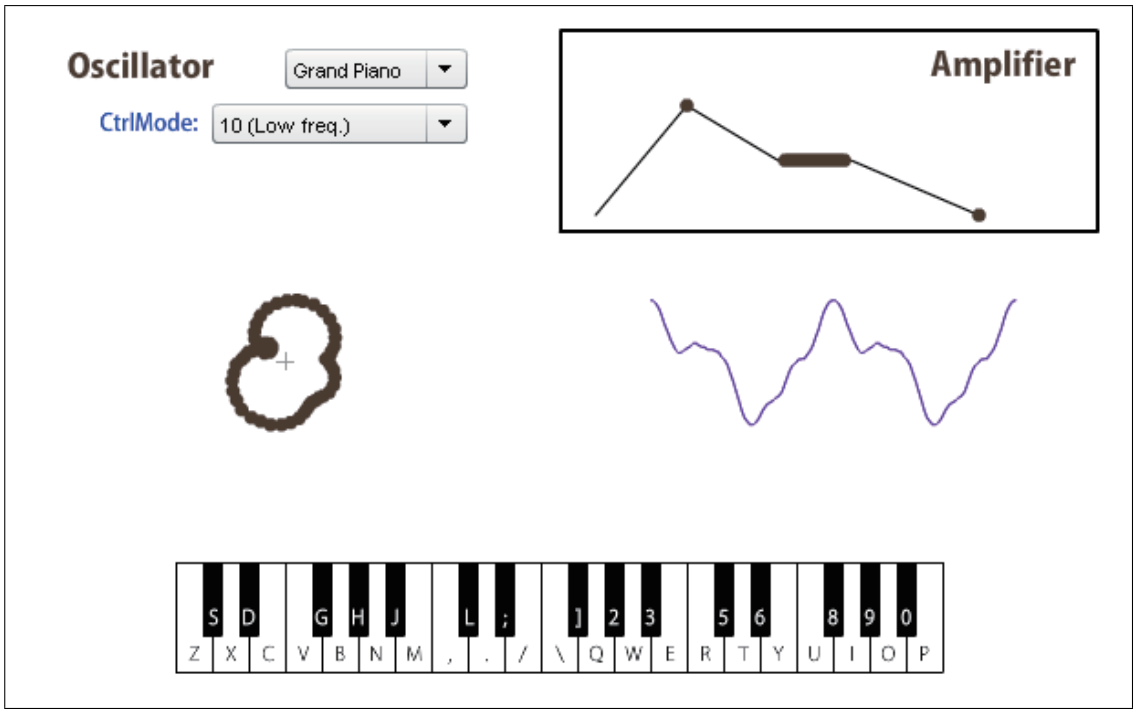


図 5.3: 開発した解析信号シンセサイザの実行画面

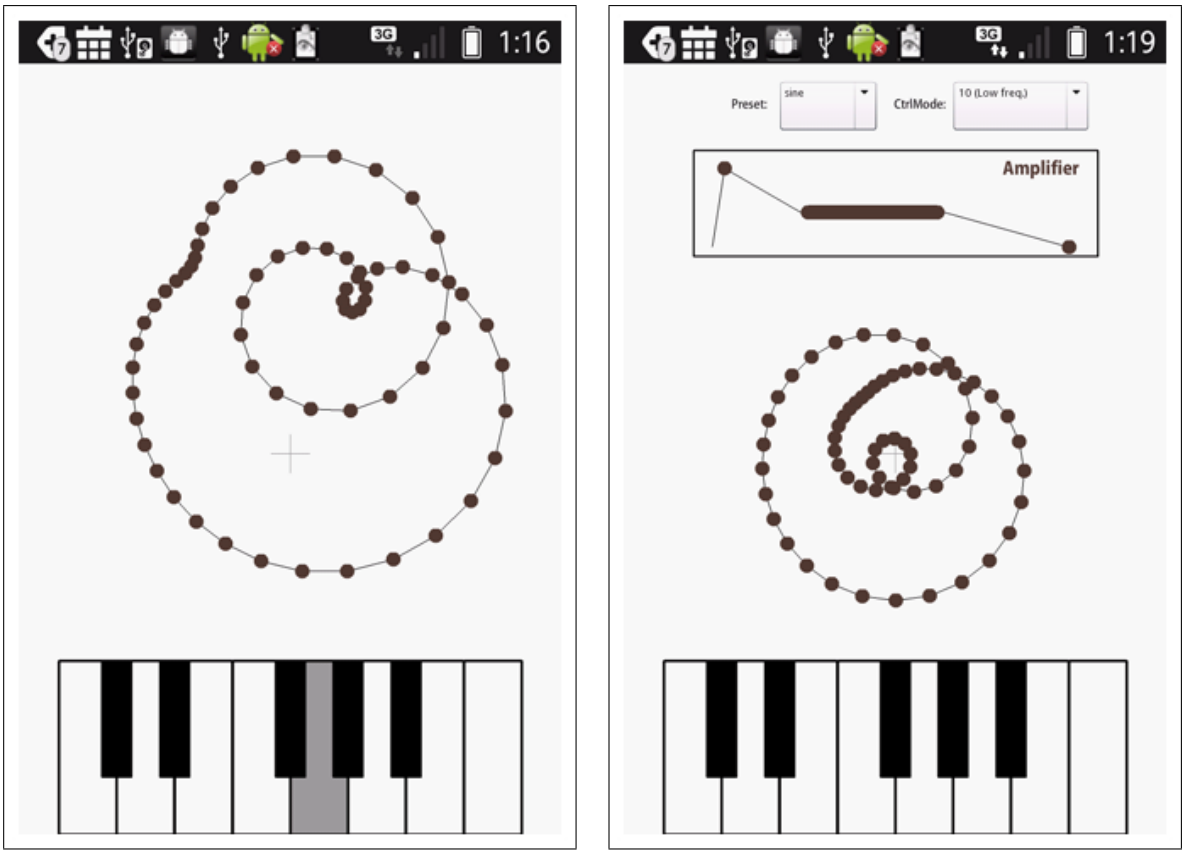
また $d = 1$ のときは基本周波数成分のみの加算となり，d が大きくなるにつれて加算される高周波数成分の割合が大きくなる．閉曲線図形の変化としては，d が大きいほど局所的な変化になる．

5.2 CloSynth

5.1.1 節で述べた手法に従って，開発した解析信号を用いる音色操作型のシンセサイザ“ CloSynth ”[21] を図 5.3 に示す．開発環境は Flash CS4，開発言語は ActionScript 3.0 である．図 5.3 の例では，実行画面中段の左の閉曲線には 64 個の制御点があり，制御点をマウス等によるドラッグによって音色の加工が可能である．制御点移動の際の印加信号に対する式 (5.2) の d の値は，10, 20, 50, 100, ∞ から選択できる．なお操作する対象である閉曲線図形としては，あらかじめ楽器音を解析信号化したものがいくつか用意されており，プルダウンメニューから読み込みができる．すなわち，既存の楽器等の音色を元に，それに図形操作によって音色に加工を加える形での音色生成が可能である．

また生成された周期音声信号に対して，時間と共に振幅の変化を加えて音として仕上げるための機能として，ADSR (Attack, Decay, Sustain, Release) エンベロープジェネレータを搭載した．生成された音色を用いた簡易的な楽器の機能として，ソフトウェアキーボードのクリック，キーボードの入力，MIDI キーボードからの入力によって簡易的な演奏を可能とした．

この解析信号シンセサイザを用いて，音色の操作を行ったところ，操作対象の既存の楽器などの音に対して「とがった音」や「キンキンした音」,「滑らかな音」などのエフェクトを加えることができた．ただし，本手法では周期信号しか扱えないため，既存の楽器音のもつハーモニックエンベロープやゆらぎなどは反映されない．これに対しては，5.4 節の方法を用いれば，楽器音全体に対



(a) 実行画面　　(b) メニュー表示時

図 5.4: CloSynth for Android の実行画面

して変形が加えれられるため，過渡的な変化を取り入れることができると考えられる．

5.3　スマートフォン版 CloSynth

タッチパネルインタフェースでの動作を実証するため，AIR for Android でスマートフォン版の CloSynth を移植実装した．Android 2.2 以上で動作する．実行に必要な apk ファイルは Android マーケットで配布している[1]（図 5.5）．実行画面のスクリーンショットを図 5.4 に示す．

PC 用の CloSynth との違いは，表示についてはスマートフォンに適するように縦型で，解析信号制御モジュールとキーボードのみを配置しており，エンベロープ機能等はメニューボタンを押したときに表示される．機能については，キー入力の範囲が 3 オクターブから 1 オクターブに減少している点である．

5.3.1　考察

SHARP 社製スマートフォン端末の IS03 で動作を確認した．画面サイズは 3.5 インチで、解像度は DVGA（640 × 960 ピクセル）である．マウスで操作する PC 用の CloSynth では制御点を正

[1] http://market.android.com/search?q=pname:air.jp.butchi.closynth

図 5.5: CloSynth for Android のダウンロード URL を示す QR コード

確にポインティングすることができたが，スマートフォンにおいては制御点のタッチ反応領域が狭くて正確にポインティングできなかった．iPad のような画面サイズが充分に大きいタブレット端末では問題なく動作することが考えられるが，画面が小さい場合はタッチ反応領域を広くするなどの工夫が必要がある．

また，画面の広さの制約上、多くのモジュールを置くことができない．そのため，メニューボタンでエンベロープ制御画面の表示・非表示を切り替えられるようにしてある．解析信号制御はシンセサイザの 1 つのモジュールとして使うことが考えられるため，どのように画面を分割し，呼び出し時にどのようなエフェクトを用いて表示するかなどのユーザインタフェースのデザインが必要となる．

実行速度について，PC 用の CloSynth ではスムーズに音色生成ができたが，CloSynth for Android では音声出力のタイムラグが大きかった．原因としてはキーボード押下時のキー検出においてループを用いているからであると考えられる．イベント制御方法を変更することによって改良できると思われる．

5.4 地表面軌道合成による拡張

Max Mathews らによって 1998-2000 年に開発・提唱された新しい物理モデル音源方式として，Scanned Synthesis がある [5] [6] ．図 5.6 のように，物理モデルで決定される動的な点群を任意の曲線で走査（サンプリング）することによって音声信号を生成する手法であり，これにより，リアルタイムに減衰などの変化をする音を生成することができる．

またこれに関連した音合成方式として，地表面軌道合成（Wave Terrain Synthesis，波形地表面合成とも呼ばれる）が提案されている [3] [4]．これは，静的な 2 変数関数（地表面）を任意の曲線で走査し，走査したそれぞれの点での関数値を波形値とする方式である．Scanned Synthesis では時間的に変化する動的な物理モデルを用いるのに対し，地表面軌道合成では時間的に固定された静的な地表面を用いる．また，Scanned Synthesis では，点群の連結が弦のように 1 次元的であったり格子状のように 2 次元的であったりと次元が自由であるのに対し，地表面軌道合成では地表面が 2 変数関数に限られる．

本節では，5.1 節で述べた解析信号を用いた音色の操作手法を，これら手法を用いて，静的な地表面を解析信号によって生成された複素信号で走査するように拡張し，それによって得られる音色のバリエーションを生成する方法について述べる．

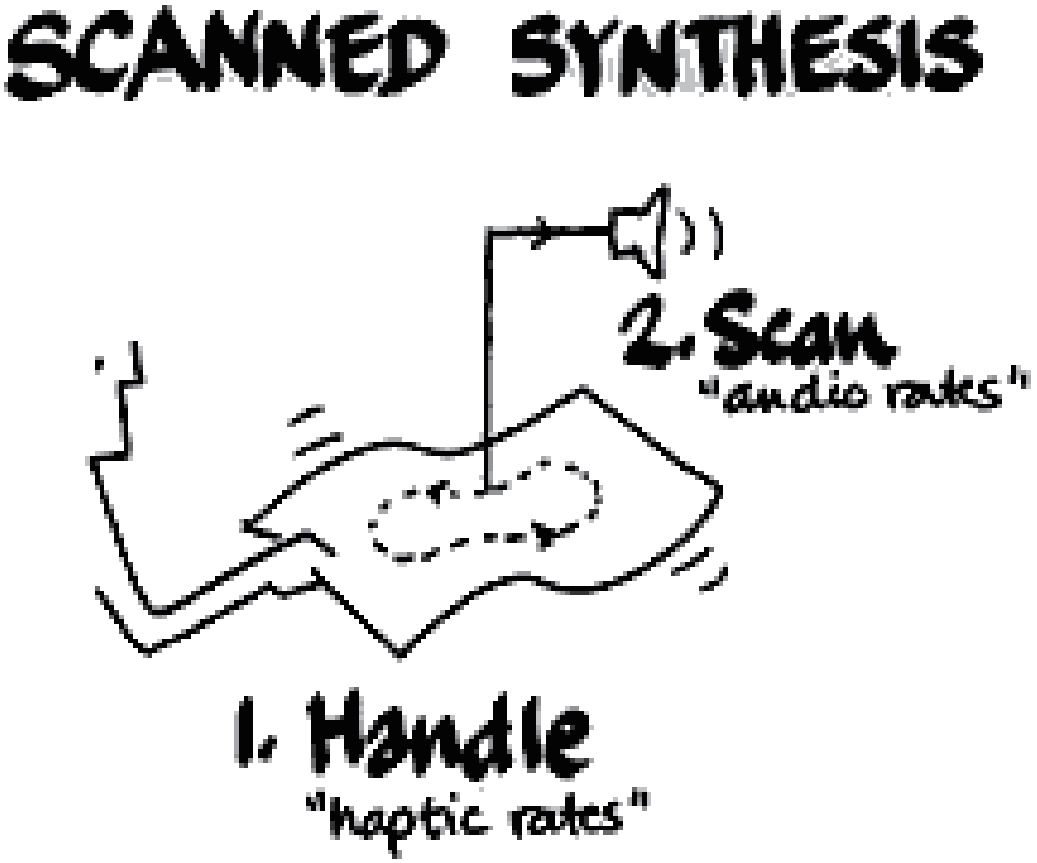


図 5.6: Scanned Synthesis [6]

5.4.1 原理

x と y を変数とする z という 2 変数関数を考え，その x に解析信号の実部を，y に解析信号の虚部を代入する．

$$x = \Re[\tilde{s}(t)] = s(t) \tag{5.3}$$

$$y = \Im[\tilde{s}(t)] = h(t) \tag{5.4}$$

例として $z = x^2 - y^2$，$s(t) = \cos(2\pi i \cdot 440t)$ と置くと，$h(t) = \sin(2\pi \cdot 440t)$，$\tilde{s}(t) = e^{2\pi i \cdot 440t}$ となり，z は式 (5.7) のようになる．

$$\begin{aligned} z &= s^2(t) - h^2(t) && (5.5)\\ &= \cos^2(2\pi \cdot 440t) - \sin^2(2\pi \cdot 440t) && (5.6)\\ &= \cos(2\pi \cdot 880t) && (5.7) \end{aligned}$$

また，z を x にすると式 (5.9) のように原信号 $s(t)$ が得られ，z を y にすると式 (5.11) のように原信号 $s(t)$ をヒルベルト変換した信号 $h(t)$ が得られる．

$$z = x \tag{5.8}$$
$$= s(t) \tag{5.9}$$

$$z = y \tag{5.10}$$
$$= h(t) \tag{5.11}$$

この z を $z = x$ とした場合は，2.1 節で述べた解析信号から音の波形を得る場合に対応することになるが，この z として別の関数を用いることで，同一の閉曲線から得られる音の波形にバリエーションを加えることができると考えられる．2.3 節で述べた閉曲線と音色との対応関係は，閉曲線の実部を求めると音色が得られるというものであったが，本節で提案する手法は Scanned Synthesis の手法を用いた拡張と考えることができる．

5.4.2 適用例と結果

この手法の例として，入力信号 $s(t)$ として正弦波，三角波，ハーモニカの音の 3 種類に対して，地表面 z として $z = y, x^2 - y^2, \sin(\pi x)\sin(\pi y)$ を与えて得られる音の波形を表 5.1 に示す．このように，地表面を変化させることによって様々な変化が生じた．z をうねりの多い関数にすることにより，高周波成分を増やすことができる．

この他に，例えばピアノの音に対して $z = \sin(\pi x)\sin(\pi y)$ とすることでギターのような音が得られるなど，音色に明らかな変化を与えることができることがわかった．

5.5 議論

本手法による音色の加工生成手法について議論を加える．

まず，従来のシンセサイザと本手法を比較する．従来のシンセサイザはパラメータ同士の関係が複雑であり，あるパラメータを変化させるときに，他のパラメータの値によっては音色が全く変化しないということもあり得る．本手法ではどの制御点を操作しても，加算される解析信号は位相が異なること以外は同じであるので，パラメータの変化を理解するのは従来のシンセサイザよりも容易であると考えられる．直感的な音色の操作が困難であるという点では，従来のシンセサイザはパラメータにラベルが付いているのみで音を出してみないと効果がわからないものが多く，これに対して本手法では制御点の移動が座標変化に直結しているので，従来よりも直感的であるといえる．多数のパラメータが必要であるという点では，本手法では 64 個の制御点があり，改善ができているとはいえず，それに加えて，本手法では複雑な音色を扱う場合は制御点に粗密が生じ，密になっている部分や交点で制御点が選択しにくい．これについての解決方法として，密になっている制御点はどれを制御しても変化はほとんど同じであるため，密になっている制御点を非表示にすることで制御点の削減が見込まれる．また，制御点が多いが，本手法における制御点は全てが連動して作用するため，まとめて 1 つのパラメータとみなすことができる．

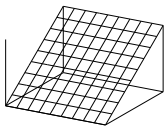
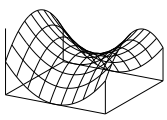
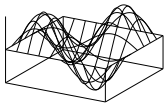
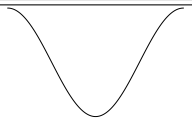
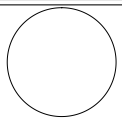
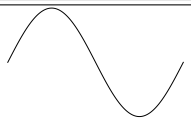
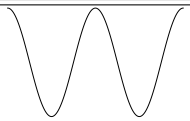
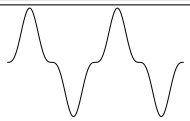
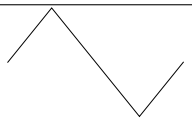
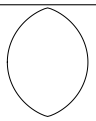
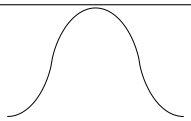
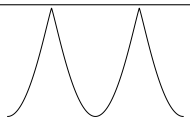
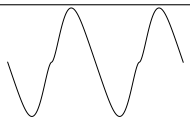
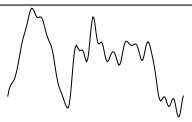
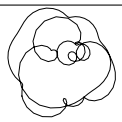
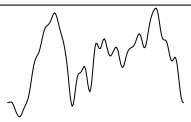
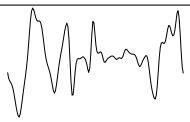
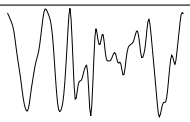
原信号 $s(t)$	$\tilde{s}(t)$	地表面 z		
				
		y	$x^2 - y^2$	$\sin(\pi x)\sin(\pi y)$
 正弦波 $\cos(2\pi \cdot 440t)$	 $e^{2\pi i \cdot 440t}$	 $\sin(2\pi \cdot 440t)$	 $\cos(2\pi \cdot 880t)$	 $\sin(\pi\cos(2\pi \cdot 440t)) + \sin(\pi\sin(2\pi \cdot 440t))$
 三角波				
 ハーモニカの音				

表 5.1: 入力信号 $s(t)$ と，それに対して地表面 z を用いて変化を加えて得られた出力信号の例

次に，解析信号シンセサイザの，音色操作の自由度・可能性について考える．著者らはこれまでに，解析信号の数学的な性質を考慮して，任意の図形を，ある程度形状を保持したまま近似的に解析信号化する手法 [1, 2] を提案したが，完全に任意の閉曲線図形を解析信号化することはできなかった．また解析信号である元の閉曲線に対して，解析信号としての性質を保ったまま，原点中心の回転と相似拡大縮小の操作は可能であるが，それ以外の例えば縦横独立の拡大縮小やせん断等の図形変形操作を行うことはできない．これに対して，本手法では，変形操作の元となる解析信号を，実際の音声信号から変換して得て，それに対して解析信号としての性質を保ったまま，制御点移動による変形操作を実現する手法である．実際に操作を行ったところ，与えられた閉曲線図形から，数分の操作練習を経て意図した図形に変形させることができた．この閉曲線図形の操作と，それに対応する音色操作との対応，すなわち音色操作の自由度や，意図した音色をどの程度生成できるかという点の定量的な評価は不十分であるものの，実際に操作をした結果から，低周波数成分を多く含む音色と高周波数成分を多く含む音色の違いを作り分けることができた．

一方，閉曲線図形とその操作を映像表現とみなすと，図形操作と音色の変化が対応づけられていることになり，新たな映像・音楽の融合の可能性とも考えられる．ただし本手法では，1 回の制御点移動の際に印加する信号が限定されるために音色の変化は乏しくなりがちであり，音楽パフォーマンスなどで利用するには面白みに欠けると思われる．この点については，制御点の移動だけでなく，それに伴って周辺にある制御点が操作指示点との距離に応じて移動するような方法等についても検討したい．

問題点として，本手法は，原理的には任意の周期信号を生成することができるが,「意図した音色」をゼロから生成するには必ずしも適していないと考えられる．ただし実用的には，音楽表現

上，既存の音色を好みの音色へと微調整したい場面は多く，本手法は従来とは異なる方法で加工することが可能である．

本手法は，閉曲線を複素平面に投影しているため，時間軸の情報がなくなっている．これによって，周波数 (音高) に依存しない音色作りが可能になっている．ただし，同じスペクトルを持つ音色も，位相が異なると閉曲線図形も変化するため，同じ音色がさまざまな形を持つ．閉曲線図形の形状情報と音色との関係については，閉曲線図形の周が自身と交差する回数が多いほど，また周に沿った閉曲線図形の回転数が多いほど，それに対応する時間軸波形が複雑になり，スペクトルの高調波成分が多くなることが予想される．また，偏角の極値数や周長などにも関係が見出されると考えられる．これらの関係については，今後解明を進める予定である．

また本手法は周期信号を作ることはできるが，波形の時間的変化などの過渡的な変化については閉曲線の操作だけでは作ることができないため，異なる操作画面を用意するか，VST のようなプラグイン化をすることが必要である．

最後に，地表面軌道合成および Scanned Synthesis を用いた本手法の拡張について考える．これらの拡張手法は，音色のバリエーションを生成することができるが，その際に用いる軌道や曲面の与え方が，得られる音に大きく影響する．そこで，例えば地表面軌道合成を適用する場合には，地表面の高さを色の濃淡で表し，ペイントソフトのブラシツールのような描画方法で地表面の高さを編集することが考えられる．また Scanned Syntehsis を適用する場合には，画面をクリック（タッチ）すると，水滴を垂らしたときのように地表面が振動し，それを解析信号で走査して音声信号をリアルタイム生成することが考えられる．これらの具体的な実装と，得られる音色との関係，およびピッチの与え方については今後検討を進めたい．

5.6 まとめ

本章では，解析信号と呼ばれる複素平面上の閉曲線図形と音声信号との対応に着目し，閉曲線に対する形状操作等の直接操作によって周期音声信号を変化させる方式による音色の操作方法を提案し，その手法によるシンセサイザを実装するとともに，解析信号に地表面軌道合成の音色合成法を適用するシミュレーションを行った．

本手法は，意図した新規の音色の作成は困難であるものの，既存の音に対する加工による新しい音色の生成には有効である．また図形と音を対応付けることにより，視覚表現と融合した音楽パフォーマンスへの応用も考えられる．

今後は，より豊かな音色の加工・生成のために，音声信号の過渡的な変化を効果的に設定する手法の検討を進め，あわせて，閉曲線図形とその操作を映像表現とみなした，新たな映像・音楽の融合の可能性を検討したい．

第6章　まとめと展望

本研究では音声と図形の変換方法として解析信号を用いる方法を提案した．これにより音声信号を曲線図形として表示することを可能にした．図形から音声への変換としては，任意の閉曲線図形をパラメータ変換することにより解析信号に近似する 2 つの手法を提案し，シミュレーションを行った．また，解析信号を拡張し，回転しても音色が不変であるような信号の条件を示した．そして，解析信号を用いたシンセサイザ “CloSynth” を提案・実装した．

音を形として見ることは，図形的な処理による音声認識や，音楽を聴く際の視覚エフェクト，VJ の演出などへの応用が考えられる．形を音として聞くことは，画像処理による音色生成への応用が考えられる．形を編集して音を編集することは，音色生成・加工のモジュールとして利用することが考えられる．ベクトル画像における曲線表現形式としてベジェ曲線が普及しているように，音声信号の図形的表現形式として解析信号が普及することを私は願っている．

謝辞

本研究の遂行にあたり懇切なご指導ご鞭撻を賜りました，金沢大学自然科学研究科教授 秋田純一博士に深く感謝の意を表します．また，有意義なるご討論，ご教示，ご助言を賜りました金沢大学自然科学研究科准教授 北川章夫博士に厚く深謝の意を表します．

大学生活を通して金沢大学大学院自然科学研究科電子情報科学専攻の先生方にはたいへんお世話になりました．

本研究を進めるにあたり，多くの有益な討論を行い，また，ご助言を頂きました，金沢大学自然科学研究科博士前期課程 2 年 上坂洋紀氏，木村俊介氏，小松原宏識氏，半田貴也氏，深田拓氏，和田智晃氏，金沢大学自然科学研究科博士前期課程 1 年 河合一樹氏，小島遼太朗氏，殿畑美也子氏，中谷優志氏，水井彩香氏，宮川裕一氏，申東源氏，金沢大学理工学域電子情報工学類 4 年 五十嵐覚氏，川上隼斗氏，川崎基輝氏，河崎泰孝氏，寺田祐司氏，中村優希氏，山本州氏，正谷智広氏に心から謝意を表します．

楽器音の音声ファイルを提供していただいた桂田先生と松山周五郎氏に感謝します．

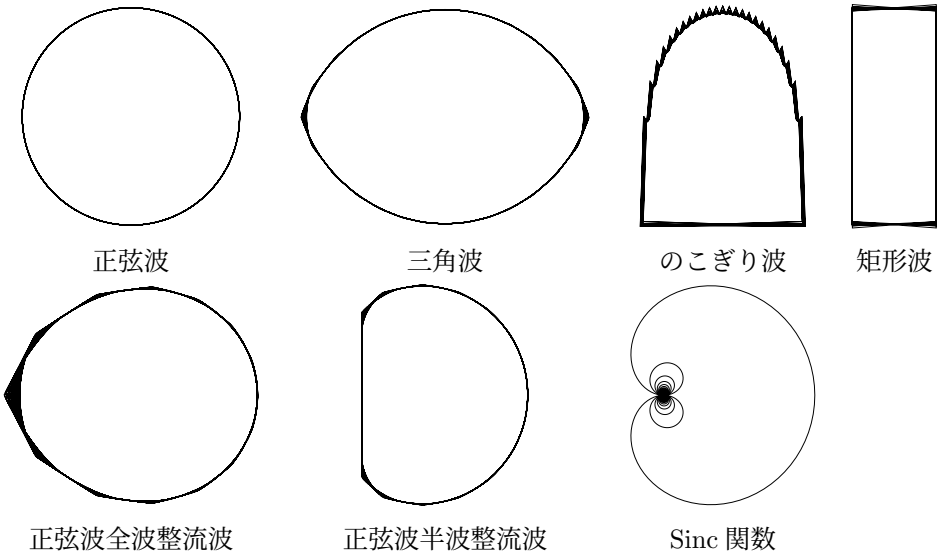
ソーシャルネットワーキングサービス mixi[1]，Facebook[2]および Twitter[3]を通し，日々の生活を暖かく応援してくれた友人の皆様に深く感謝いたします．

最後に，進学を支持して頂き，生活を支えてくれる両親，そして日頃から応援してくれる姉に厚く御礼申し上げます．

[1] http://mixi.jp/
[2] http://www.facebook.com/
[3] http://twitter.com/

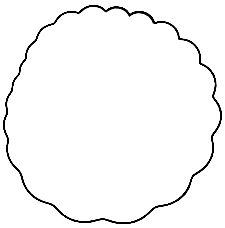
付 録A　楽器音などの解析信号

A.1　プリミティブな音

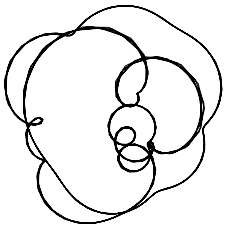


A.2 MIDI音

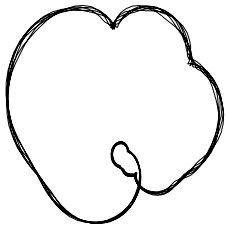
MIDI 音は Microsoft GS Wavetable SW Synth のものを利用した.



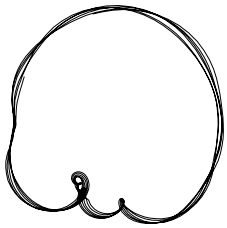
005 El. Piano



023 Harmonica

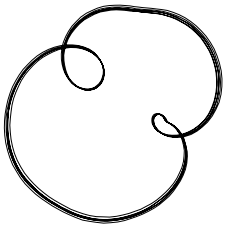


025 Nylon Guitar

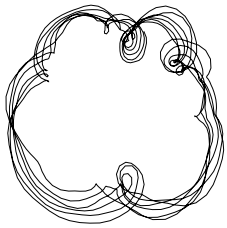


027 Jazz Guitar

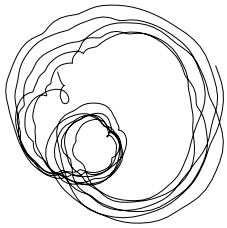
041 Violin



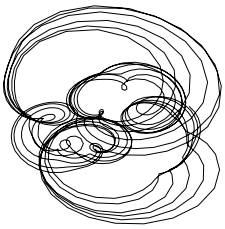
047 OrchestraHarp



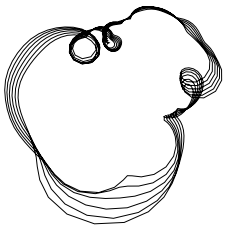
049 Strings 1



053 Choir Aahs



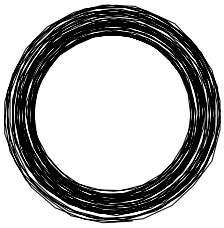
081 Square Lead

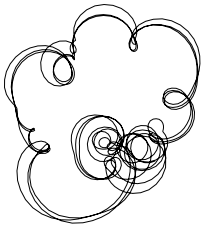


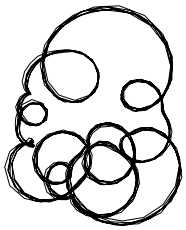
082 Saw Lead

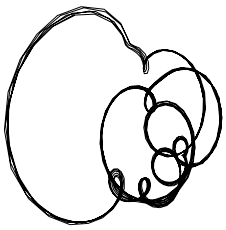
A.3　楽器音

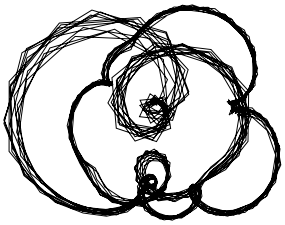
桂田祐史氏により提供された音声ファイルを用いた.

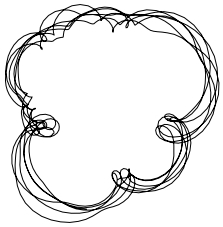

チェレスタ

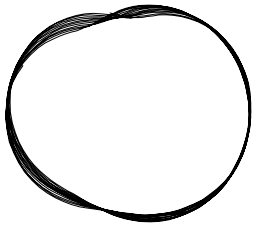

ギター

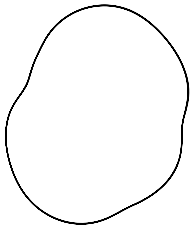

ギター（高）

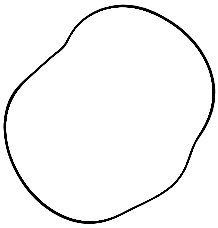

ハーモニカ

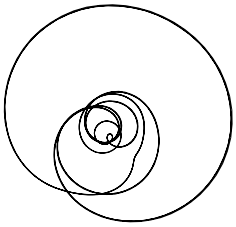

ハーモニカ（高）

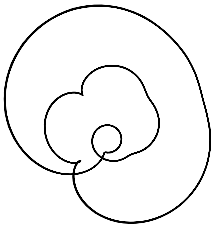

ピアノ

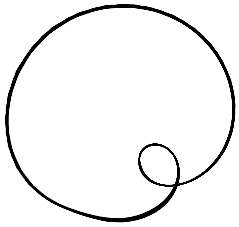

ピアノ（高）

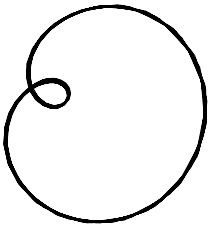

リコーダー

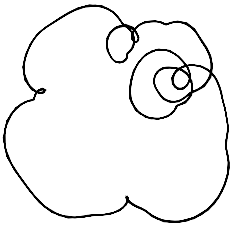

リコーダー（高）

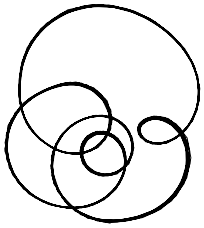

三味線


三味線（高）


トランペット

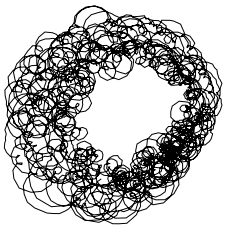

トランペット（高）


ヴァイオリン

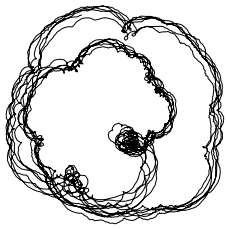

ヴァイオリン（高）

A.4　声

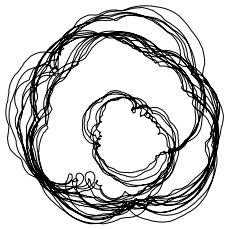
あ



い

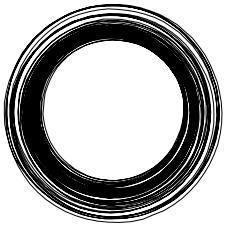


う

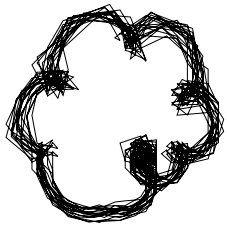


え

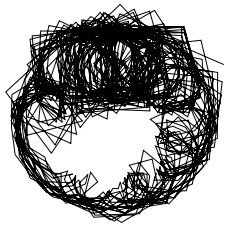
お



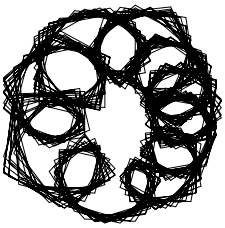
口笛



ホーミー 1



ホーミー 2



ホーミー 3

付 録B　プログラム

プログラムは以下のページに掲載した.
http://butchi.jp/documents/d-thesis/
ミラー: http://merl.ec.t.kanazawa-u.ac.jp/closynth/

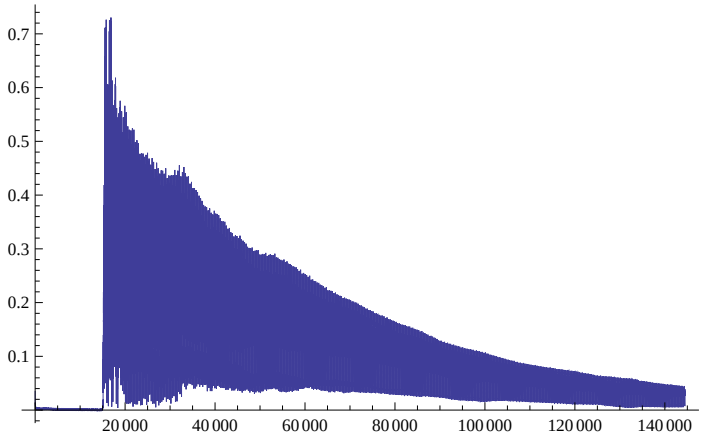
B.1　Wave ファイルの解析信号をプロットするプログラム

動作環境: Mathematica 8.0

```
(*離散ユニットステップフィルタ*)

step[n_Integer?Positive]:=

1 + Sign[Sin[πRange[1, −1, −2/n]//Most]]

(*実関数の複素化*)

wind[li_List]:=InverseFourier[Fourier[li] * step[Length[li]]]


wavname = "guitar";

wavfile = Import[wavname <> ".wav"];

li = wavfile[[1]][[1]][[1]];

len = li//Length

compli = wind[li];
```

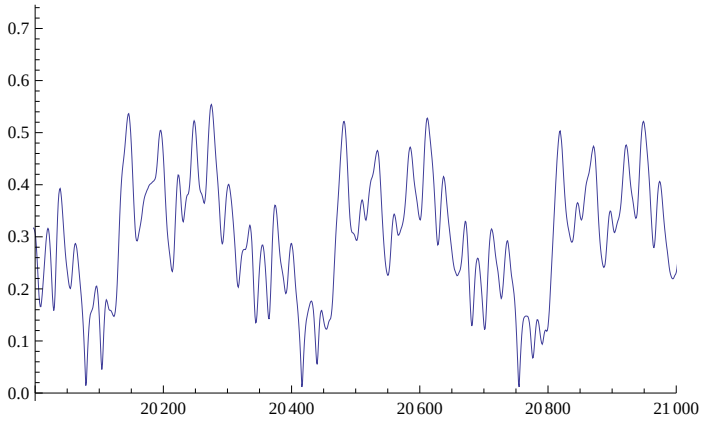
144567

```
ListLinePlot[Abs[compli], PlotRange → {All, All}]
```

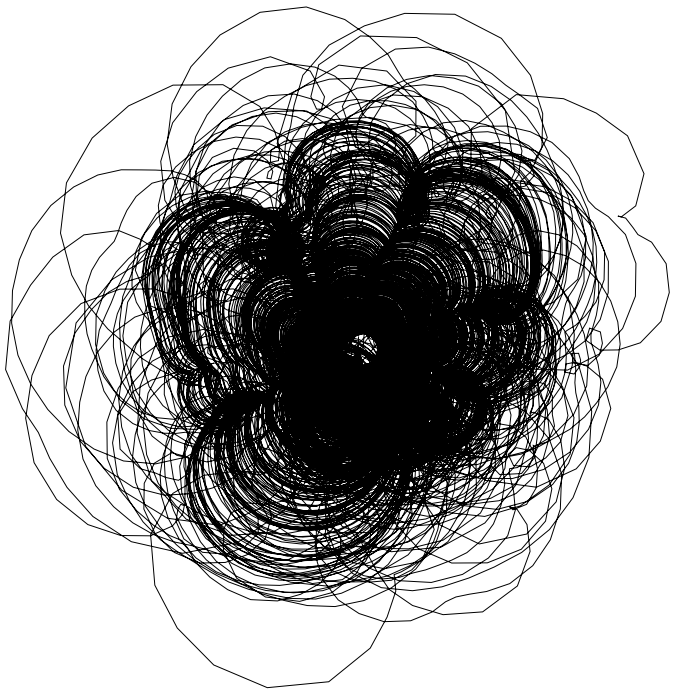



```
mainrange = {20000, 21000}

ListLinePlot[Abs[compli], PlotRange → {mainrange, All}]
```

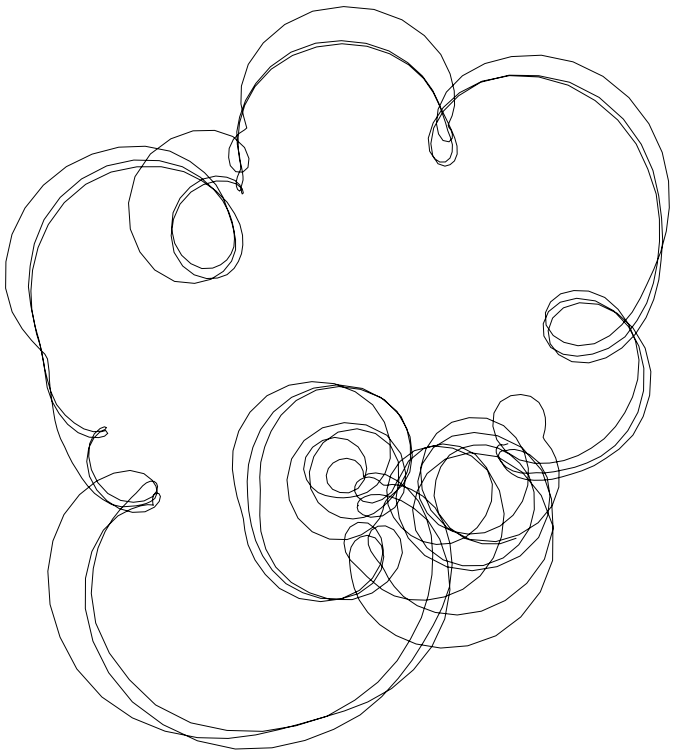
$\{20000, 21000\}$




```
ListLinePlot[{Re[compli], Im[compli]}T, PlotRange → All,

AspectRatio → Automatic, Axes → False, PlotStyle → Black]
```



```
ListLinePlot[Take[{Re[compli], Im[compli]}T, mainrange],
PlotRange → All, AspectRatio → Automatic, Axes → None,
PlotStyle → Black]
```



B.2 Wave ファイルから解析信号化した Wave ファイルを出力するプログラム

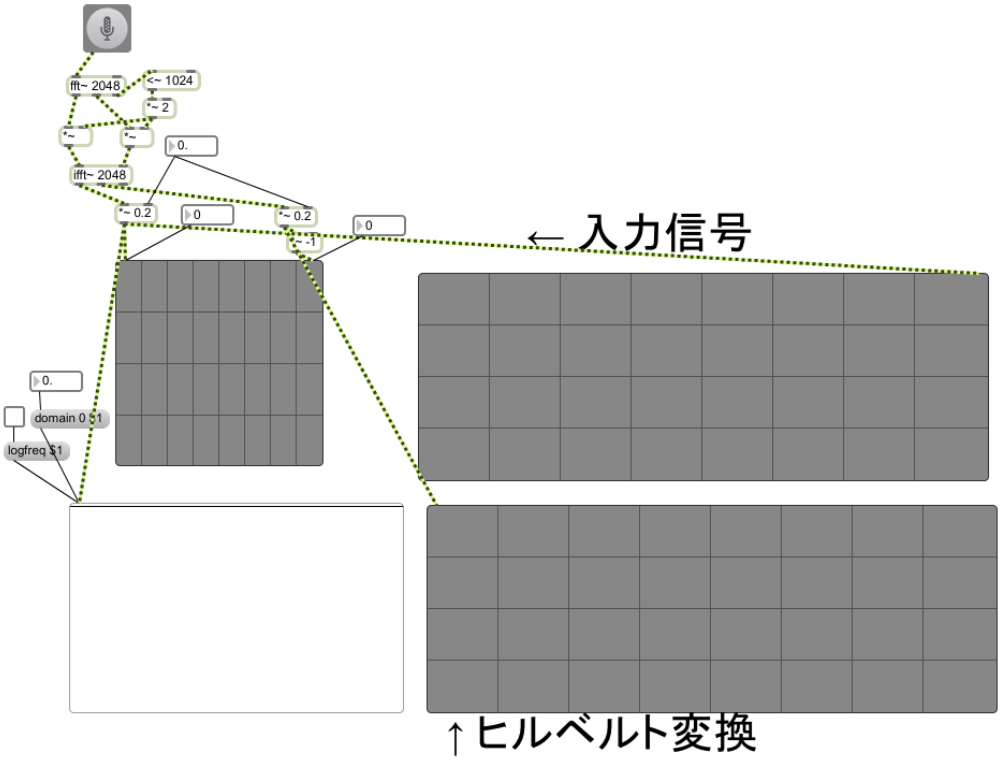
元の Wave ファイルのモノラル音声（L チャンネル）をヒルベルト変換（R チャンネル）し，ステレオ音声としてファイル出力するプログラムを示す.
動作環境: MATLAB R2009a

```
clear
% フォルダまるごと複素ステレオ変換 ('*.wav' →'*.wave')
sndpath='C:\sound\';
snd=dir([sndpath '*.wav']);
if(isempty(snd))
    'No wave files.'
end
for i=1:length(snd)
    snd(i).name
    if(length(dir([sndpath snd(i).name 'e']))==1)
        'transformed file is exsist.'
        continue;
    end
    s=importdata([sndpath snd(i).name]);
    len=length(s.data);
    comps=ifft(fft(s.data(:,1)).*[1 2*ones(1,ceil(len/2)-1) ones(1,mod(len+1,2)) zeros(1,ceil(len/2)-1)]');
    % plot(comps);
    s2=[real(comps) imag(comps)];
    % sound(s2,s.fs)
```

```
    wavwrite(s2,s.fs, [sndpath snd(i).name 'e']);
end
```

B.3　リアルタイムに解析信号をプロットするプログラム

動作環境: Max 5



B.4　拡張した解析信号をプロットするプログラム

動作環境: Mathematica 8.0

(*離散ユニットステップフィルタ*)

step[n_Integer?Positive]:=1 + Sign[Sin[πRange[1, −1, −2/n]//Most]]

(*実関数の複素化*)

wind[li_List]:=

InverseFourier[Fourier[li, FourierParameters → {1, −1}] ∗ step[Length[li]],

FourierParameters → {1, −1}]

repeat[li_, len_]:=PadRight[{}, len, li]

(*li を補充して長さを len にする*)

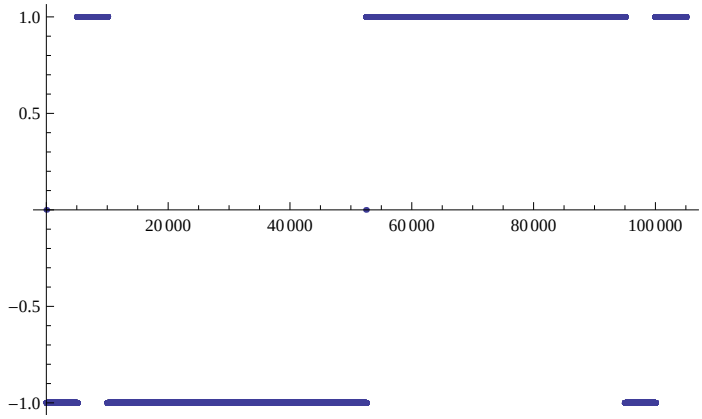
repeat[li_, len_]:=PadRight[{}, len, li]

(*0 からスタートし、終点の値を始点の値にする*)

```
periodiclistlineplot[li_]:=
ListLinePlot[{Range[0, Length[li]], Append[#, First[#]]&[li]}⊤, Axes → None,
PlotStyle → {Black, Thickness[0.005]}]
(* 複素平面のプロット、始点と終点を繋げる *)
complexlistlineplot[li_]:=
ListLinePlot[Append[#, First[#]]&[{Im[#], Re[#]}&/@li], AspectRatio → Automatic,
Axes → None, PlotStyle → {Black, Thickness[0.005]}]
swap[li_List, x_ ↔ y_]:=ReplacePart[li, {x → li[[y]], y → li[[x]]}]
swap[li_List, pairs : {(_ ↔ _)...}]:=Fold[swap, li, pairs]
extendedStep[n_, {x___, 0, y___}]:=extendedStep[n, {x, y}]
extendedStep[n_, swapList_List]:=swap[step[n], # + 1 ↔ n − # + 1&/@swapList]
extendedWind[li_List]:=wind[li]
extendedWind[li_List, swapList_List]:=
 InverseFourier[Fourier[li, FourierParameters → {1, −1}]∗
extendedStep[Length[li], swapList], FourierParameters → {1, −1}]

wavname = "023";
wavfile = Import[wavname <> ".wav"];
li = wavfile//First//First//First;
rate = SampleRate → (wavfile//First//Last)
len = li//Length
compli = wind[li];
swapList = Range[5000, 10000];
extendedCompli = extendedWind[li, swapList];
ListPlot[
Im@
Fourier[Im@InverseFourier[extendedStep[len, swapList], FourierParameters → {1, −1}],
FourierParameters → {1, −1}], PlotRange → All]
(*拡張したフィルタ*)
```

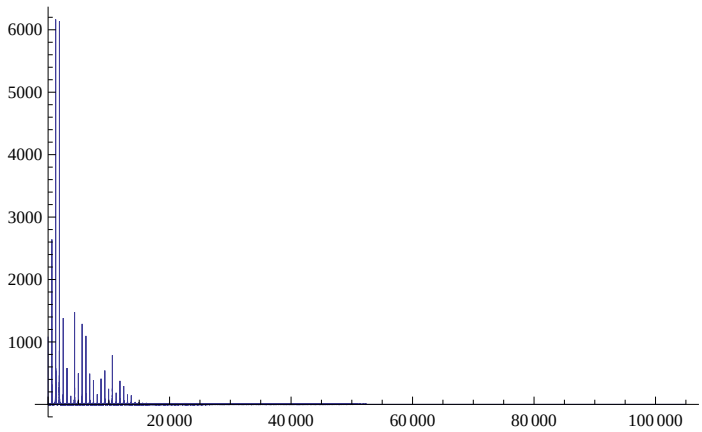
```
SampleRate → 44100
```

```
104958
```

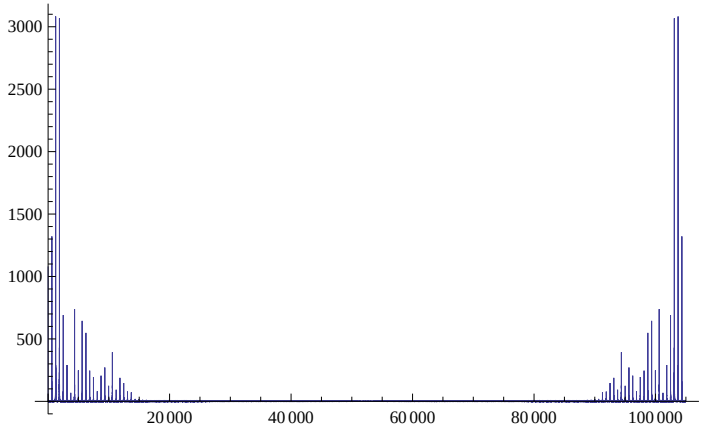



```
spectrumPlot[li_]:=ListLinePlot[Fourier[li, FourierParameters → {1, −1}]//Abs,
PlotRange → All]
```

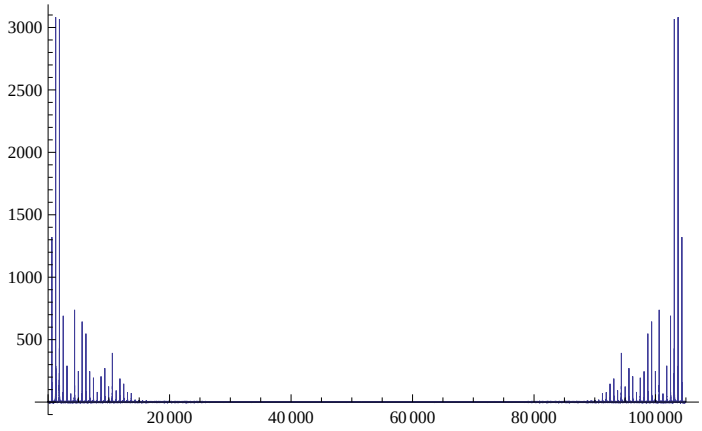
```
spectrumPlot[compli]
```



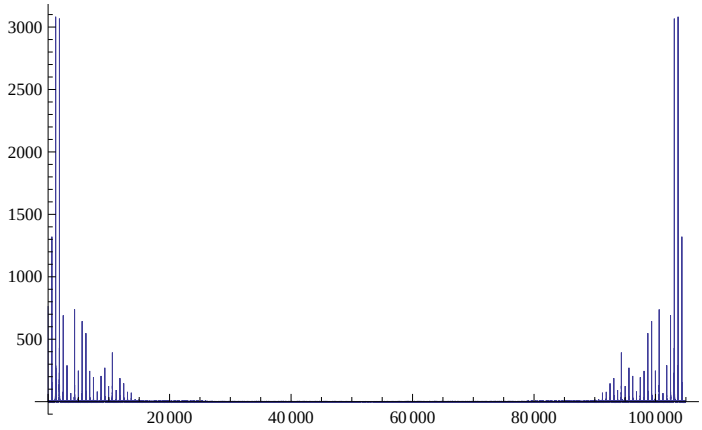

```
spectrumPlot[Re[compli]]
```



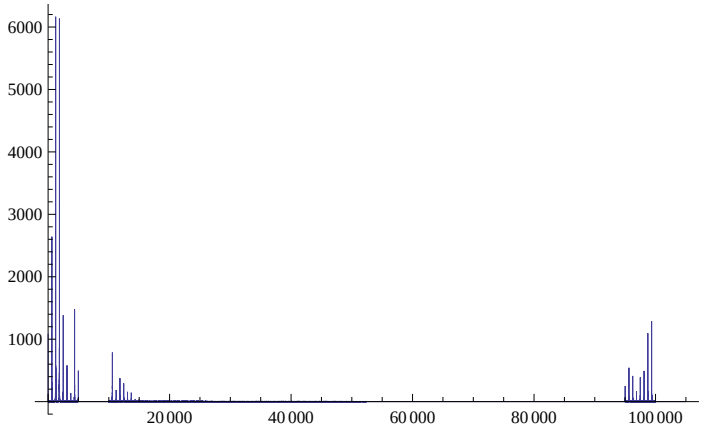

```
spectrumPlot[Im[compli]]
```



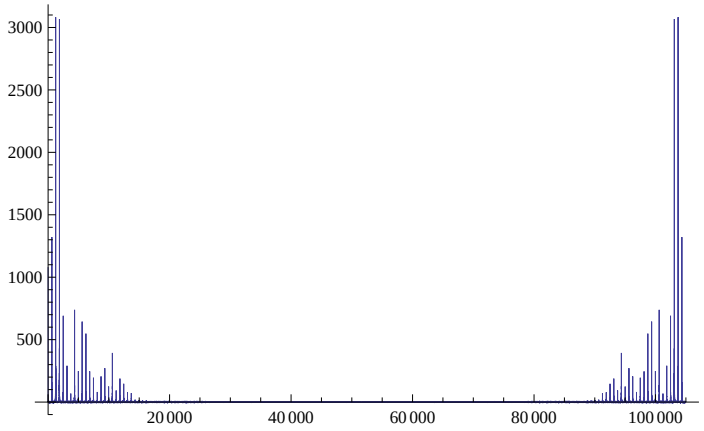
spectrumPlot $\left[\text{Re}\left[e^{i\frac{\pi}{4}}\,\text{compli}\right]\right]$



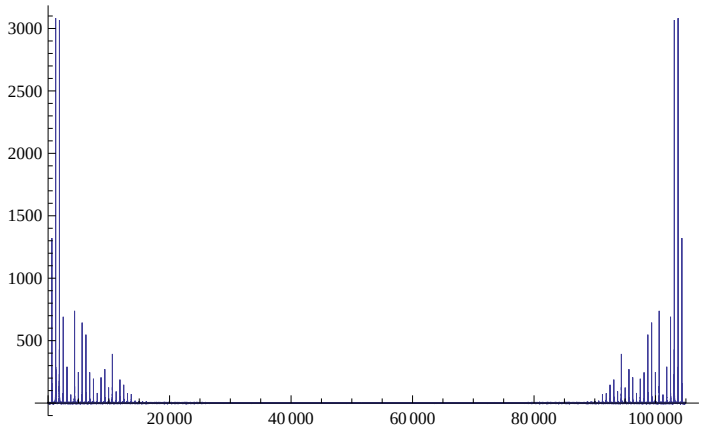
spectrumPlot[extendedCompli]



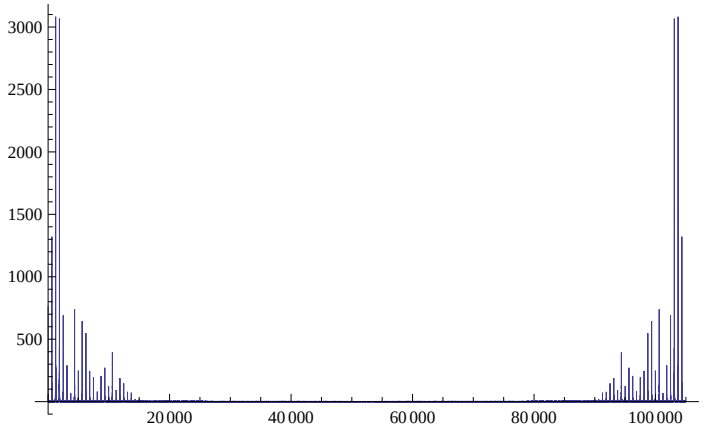
spectrumPlot[Re[extendedCompli]]



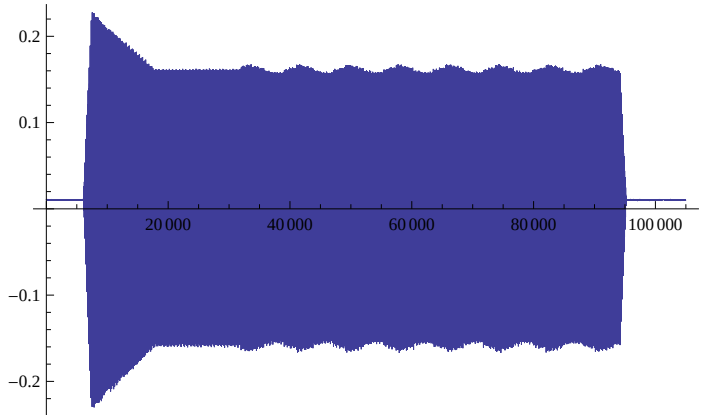
spectrumPlot[Im[extendedCompli]]



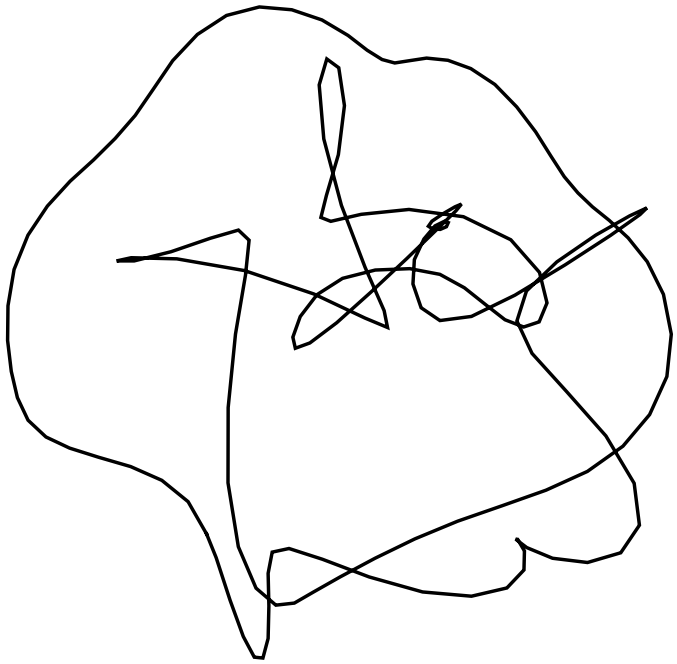
spectrumPlot $\left[\text{Re}\left[e^{i\frac{\pi}{4}}\,\text{extendedCompli}\right]\right]$



ListLinePlot[li]




```
mainRange = {20000, 20168};

periodiclistlineplot[Re[Take[compli, mainRange]]]

complexlistlineplot[Take[compli, mainRange]]

periodiclistlineplot[Re[Take[extendedCompli, mainRange]]]

complexlistlineplot[Take[extendedCompli, mainRange]]
```



B.5　ランダムな解析信号から曲線の特徴量を求めるプログラム

動作環境: Mathematica 8.0

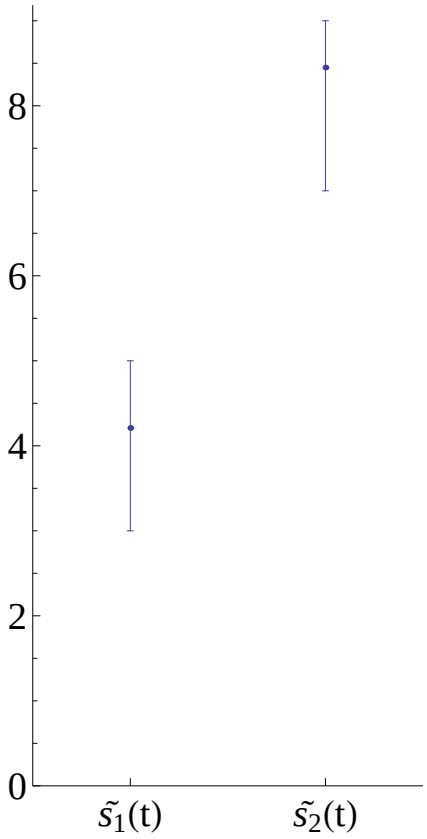
```
(*再実行しても結果が同じになるように*)
SeedRandom[0]

numberOfExtremum[li_List]:=
Length[Select[Differences[li] * RotateRight[Differences[li]], # ≤ 0&]]

(*線分が交わるかどうかを求める*)
normalize[p_, a_, b_]:=If [b - a ≠ 0, (p-a)/(b-a), ComplexInfinity]
cross0Q::usage = "実数区間 [x,y] が原点を含むか、ただし x==y のときは False";
cross0Q[x_, y_]:=If[x ≠ y, IntervalMemberQ[Interval[{x, y}], 0], False]
(*(x ≥ 0&&y ≤ 0)||(x ≤ 0&&y ≥ 0)*)
intervalIntersectionQ[a_, b_]:=!((a < 0&&b < 0)||(a > 1&&b > 1))
complexToVector[x_]:={Re[x], Im[x]}
cross01Q::usage = "入力された 2 つの複素数で描かれるが線分 (0+0i,1+0i) と交わるか";
cross01Q[p2_, q2_]:=If [cross0Q[Im[p2], Im[q2]]&&0 ≤ Re [(Im[p2]q2-Im[q2]p2)/(Im[p2]-Im[q2])] ≤ 1,
True, If[Im[p2] == 0&&Im[q2] == 0, intervalIntersectionQ[Im[p2], Im[q2]], False]]
crossQ::usage = "線分 (p,q) が線分 (a,b) と交わるか";
crossQ[{{p_, q_}, {a_, b_}}]:=
```

```
If[a == b, If[p == q, a == p, cross01Q[normalize[a, p, q], normalize[b, p, q]]],
cross01Q[normalize[p, a, b], normalize[q, a, b]]]
curveCrossQ[{{p_, q_}, {a_, b_}}]:=If[q ≠ a&&b ≠ p, crossQ[{{p, q}, {a, b}}], False]

n = 10;
norList1 = {};
noargeList1 = {};
noabseList1 = {};
noreList1 = {};
lenList1 = {};
noiList1 = {};
Do[
(*amp = Table [Random[]/k, {k, 1, n}] ;*)
amp = {1, 1, 1, 1, 1, 0, 0, 0, 0, 0};
rot = Table[2πRandom[], {n}];
(*rot = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}*)
c[t_]:= ∑_{k=1}^{n} amp[[k]]Exp[ik(t - rot[[k]])];

(*NumberOfRotation*)
dt = 2π/100;
argList = Table[Arg[c[t + dt] - c[t]], {t, 0, 2π, dt}];
 nor = Length[Select[Differences[argList], # ≤ -π&]]-
Length[Select[Differences[argList], # ≥ π&]]; (*これが回転数*)
AppendTo[norList1, nor];

(*NumberOfArgExtremum*)
dt = 2π/200;
argList = Table[Arg[c[t]], {t, 0, 2π, dt}];
 correctionArgList =
 Rest[argList]+
Accumulate[Piecewise[{{-2π, # ≥ π}, {2π, # ≤ -π}}]&/@Differences[argList]];
noarge = numberOfExtremum[correctionArgList]/2;
AppendTo[noargeList1, noarge];

(*NumberOfAbsExtremum*)
dt = 2π/200;
```

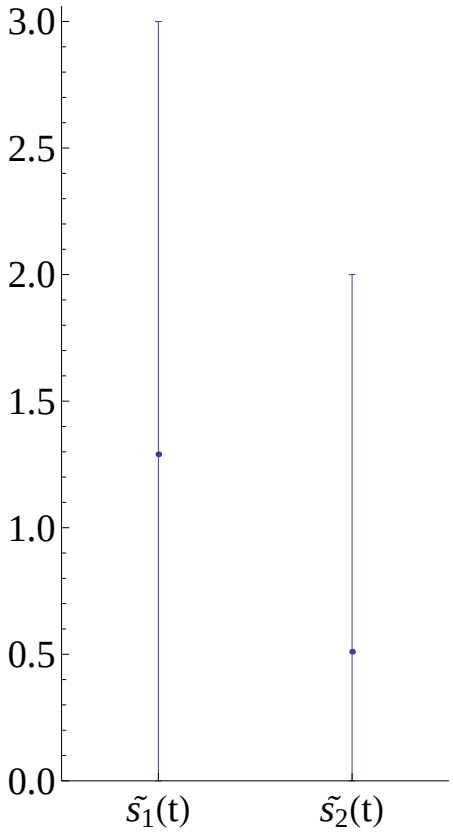
```
absList = Table[Abs[c[t]], {t, 0, 2π, dt}];
 correctionAbsList =
 Rest[absList]+
Accumulate[Piecewise[{{-2π, # ≥ π}, {2π, # ≤ -π}}]&/@Differences[absList]];
noabse = numberOfExtremum[correctionAbsList]/2;
AppendTo[noabseList1, noabse];

(*NumberOfReExtremum*)
dt = 2π/200;
reList = Table[Re[c[t]], {t, 0, 2π, dt}];
 correctionReList =
 Rest[reList]+
Accumulate[Piecewise[{{-2π, # ≥ π}, {2π, # ≤ -π}}]&/@Differences[reList]];
nore = numberOfExtremum[correctionReList]/2;
AppendTo[noreList1, nore];

(*Length*)
dt = 2π/500;
curveLength = Sum[Abs[c[t + dt] - c[t]], {t, 0, 2π, dt}];
AppendTo[lenList1, curveLength];

(*NumberOfIntersection*)
dt = 2π/200;
li = Most[Table[c[t], {t, 0, 2π, dt}]];
lines = {li, RotateLeft[li]}T;
noi = Count[curveCrossQ/@Subsets[lines, {2}], True];
AppendTo[noiList1, noi];
, {100}]

n = 10;
norList2 = {};
noargeList2 = {};
noabseList2 = {};
noreList2 = {};
lenList2 = {};
noiList2 = {};
```

```
Do[
(*amp = Table [Random[]/k, {k, 1, n}] ;*)
amp = {0, 0, 0, 0, 0, 1, 1, 1, 1, 1};
rot = Table[2πRandom[], {n}];
(*rot = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}*)
c[t_]:= Σ_{k=1}^{n} amp[[k]]Exp[ik(t - rot[[k]])];

(*NumberOfRotation*)
dt = 2π/100;
argList = Table[Arg[c[t + dt] - c[t]], {t, 0, 2π, dt}];
 nor = Length[Select[Differences[argList], # ≤ -π&]]-
Length[Select[Differences[argList], # ≥ π&]]; (*これが回転数*)
AppendTo[norList2, nor];

(*NumberOfArgExtremum*)
dt = 2π/200;
argList = Table[Arg[c[t]], {t, 0, 2π, dt}];
 correctionArgList =
 Rest[argList]+
Accumulate[Piecewise[{{-2π, # ≥ π}, {2π, # ≤ -π}}]&/@Differences[argList]];
noarge = numberOfExtremum[correctionArgList]/2;
AppendTo[noargeList2, noarge];

(*NumberOfAbsExtremum*)
dt = 2π/200;
absList = Table[Abs[c[t]], {t, 0, 2π, dt}];
 correctionAbsList =
 Rest[absList]+
Accumulate[Piecewise[{{-2π, # ≥ π}, {2π, # ≤ -π}}]&/@Differences[absList]];
noabse = numberOfExtremum[correctionAbsList]/2;
AppendTo[noabseList2, noabse];

(*NumberOfReExtremum*)
dt = 2π/200;
reList = Table[Re[c[t]], {t, 0, 2π, dt}];
```

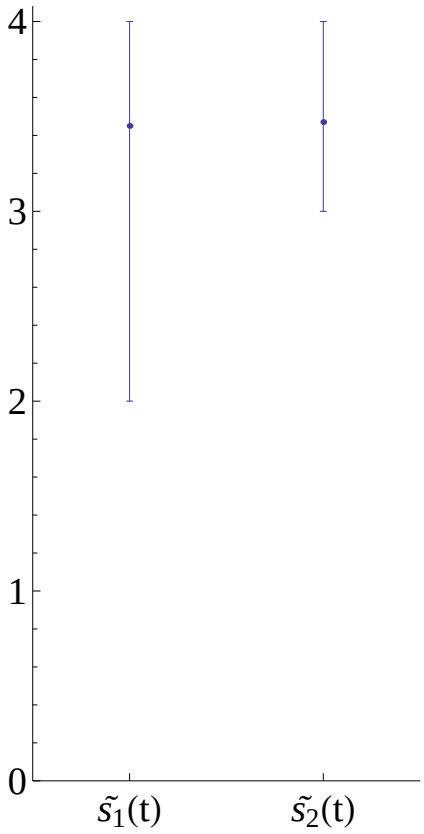
```
correctionReList =
 Rest[reList]+
Accumulate[Piecewise[{{-2π, # ≥ π}, {2π, # ≤ -π}}]&/@Differences[reList]];
nore = numberOfExtremum[correctionReList]/2;
AppendTo[noreList2, nore];

(*Length*)
dt = 2π/500;
curveLength = Sum[Abs[c[t + dt] - c[t]], {t, 0, 2π, dt}];
AppendTo[lenList2, curveLength];

(*NumberOfIntersection*)
dt = 2π/200;
li = Most[Table[c[t], {t, 0, 2π, dt}]];
lines = {li, RotateLeft[li]}⍉;
noi = Count[curveCrossQ/@Subsets[lines, {2}], True];
AppendTo[noiList2, noi];

, {100}]

Needs["ErrorBarPlots"]

resultPlot[list1_, list2_]:=
ErrorListPlot[
{{{1, Mean[list1]}, ErrorBar[{Min[list1] - Mean[list1], Max[list1] - Mean[list1]}]},
{{2, Mean[list2]}, ErrorBar[{Min[list2] - Mean[list2], Max[list2] - Mean[list2]}]}},
PlotRange → {{0.5, 2.5}, {0, All}}, AspectRatio → 2,
 Ticks → {{{1, "s̃₁(t)"}, {2, "s̃₂(t)"}}}, LabelStyle → (FontSize → 20)]

resultPlot[norList1, norList2]
```



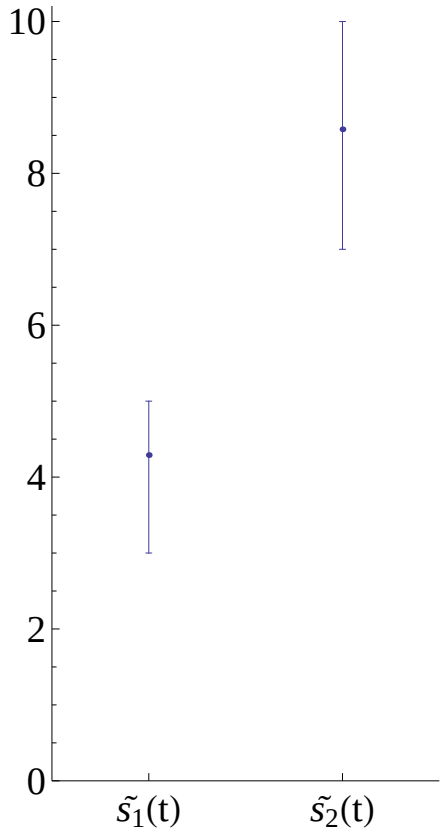
resultPlot[noargeList1, noargeList2]



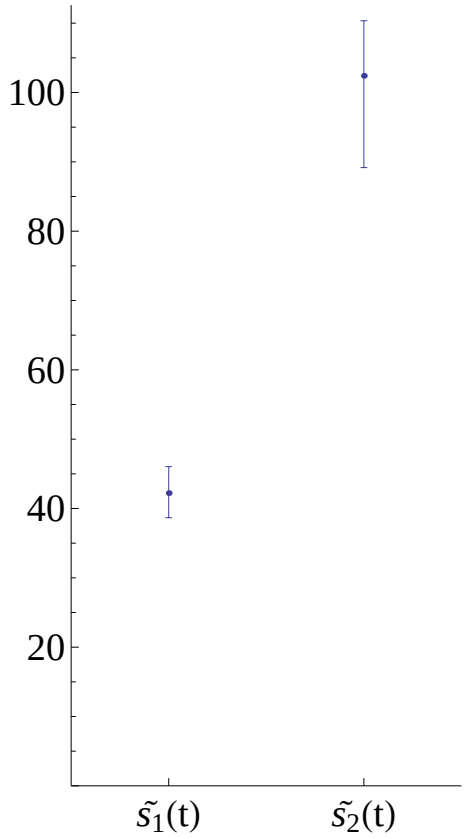
resultPlot[noabseList1, noabseList2]



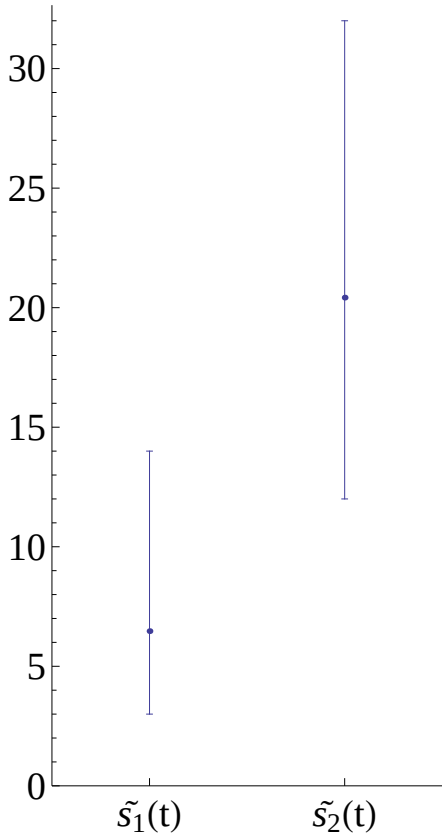
resultPlot[noreList1, noreList2]



resultPlot[lenList1, lenList2]



resultPlot[noiList1, noiList2]



B.6 Mathematica 版 CloSynth

動作環境: Mathematica 8.0（音声は出ない）

```
(*createdat2012/3/11*)
(*fromAnalyticSignalSynthesizer.nb*)

(*離散ユニットステップフィルタ*)
step[n_Integer?Positive]:=1 + Sign[Sin[πRange[1, −1, −2/n]//Most]]
(*実関数の複素化*)
wind[li_List]:=InverseFourier[Fourier[li] * step[Length[li]]]

Manipulate[DynamicModule[{s = {Re[#], Im[#]}&/@N[Exp[−2πiMost[Range[0, 1, 1/64]]]]},
Grid[
{{Graphics[{{EdgeForm[{Black}], FaceForm[], Polygon[Dynamic[s]]},
Table[With[{k = k}, Locator[Dynamic[s[[k]], (tmp = # − s[[k]]; s = s + ({Re[#], Im[#]}&
  /@wind [RotateRight [((1+Cos[2πMost[Range[0,1,1/64]]])/2)^a Complex@@tmp, k − 1]])) &]], {k, 1, 64}]},
PlotRange → 1.5, ImageSize → 500], SpanFromLeft},
{Dynamic[ListLinePlot[Reverse/@s, AspectRatio → Automatic, PlotRange → All, Mesh → All]],
```

```
Dynamic[ListPlot[{Range[0, 63], Abs[Fourier[Complex@@@s]]}T,
PlotRange → {{−1, 15}, {0, All}}, Filling → Axis]], Dynamic[ListLinePlot[sT//First, Mesh → All]]}]],
{{a, 10}, 1, 100}]
```

B.7 Flash 版 CloSynth

Flash CS4 上で別途ムービークリップおよびコンポーネントを配置した.

```
var n:int=64;
function mult(a:Array,b:Array):Array{
  return [a[0]*b[0]-a[1]*b[1] , a[0]*b[1]+a[1]*b[0]];
}

//印加信号定義
var pulsereal0_array:Array = new Array(n);
var pulseimag0_array:Array = new Array(n);
for(var i=0; i<n; i++){pulsereal0_array[i]=1}
for(i=0; i<n; i++){pulseimag0_array[i]=0}
var pulsereal10_array:Array = new Array(n);
var pulseimag10_array:Array = new Array(n);
pulsereal10_array = [1,0.9762,0.908,0.8044,0.6784,0.5441,0.4147,0.2998,0.2053,0.1328,0.081,0.0465,0.0249,0.0
pulseimag10_array = [0,-0.1702,-0.3245,-0.4498,-0.5377,-0.5859,-0.5973,-0.5789,-0.5397,-0.4888,-0.434,-0.380
];
var pulsereal20_array:Array = new Array(n);
var pulseimag20_array:Array = new Array(n);
pulsereal20_array = [1,0.9529,0.8244,0.6471,0.4602,0.2961,0.1719,0.0899,0.0421,0.0176,0.0066,0.0022,0.0006,0
pulseimag20_array = [0,-0.2384,-0.4336,-0.5572,-0.6032,-0.5854,-0.5283,-0.4556,-0.3846,-0.3239,-0.2753,-0.23
var pulsereal50_array:Array = new Array(n);
var pulseimag50_array:Array = new Array(n);
pulsereal50_array = [1,0.8865,0.6171,0.3368,0.1437,0.0477,0.0123,0.0024,0.0004,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
pulseimag50_array = [0,-0.3608,-0.5724,-0.5996,-0.5134,-0.4023,-0.3126,-0.251,-0.2092,-0.1794,-0.1566,-0.138
var pulsereal100_array:Array = new Array(n);
var pulseimag100_array:Array = new Array(n);
pulsereal100_array = [1,0.7858,0.3808,0.1134,0.0206,0.0023,0.0002,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
pulseimag100_array = [0,-0.4723,-0.6071,-0.4891,-0.3443,-0.2527,-0.1996,-0.1655,-0.1412,-0.1227,-0.1079,-0.0
var pulserealinf_array:Array = new Array(n);
var pulseimaginf_array:Array = new Array(n);
pulserealinf_array = [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
pulseimaginf_array = [0,-0.6361,0,-0.2107,0,-0.1248,0,-0.0873,0,-0.0661,0,-0.0521,0,-0.0421,0,-0.0345,0,-0.0
var pulsereal_array:Array=pulsereal10_array;
var pulseimag_array:Array=pulseimag10_array;

var ygraph:Shape=new Shape(); // 縦の時間軸プロット場
ygraph.x=z_mc.x+200;
ygraph.y=z_mc.y;
addChild(ygraph);
var zgraph:Shape=new Shape(); // 解析信号の複素平面プロット場
zgraph.x=z_mc.x;
zgraph.y=z_mc.y;
addChild(zgraph);
```

```
var phase:Number=0;
const PI2:Number=Math.PI*2;
var soundChannelObject:SoundChannel = new SoundChannel();
function ASSoundPlay() {
  var soundObj:Sound = new Sound();
  //まずは SAMPLE_DATA イベントにリスナー登録
  soundObj.addEventListener(SampleDataEvent.SAMPLE_DATA,onSampleData);
  //mp3 などの音声を割り当てずにそのまま play() で再生開始すると、
  //音のバッファが空になるたびに SAMPLE_DATA イベントが発生します。
  soundChannelObject=soundObj.play();
}
var t:int=0;//絶対サンプル数
var xval=0, yval=0;
var freq;
function onSampleData(e:SampleDataEvent):void {
  //一度のイベントごとに 2048〜8192 サンプル×左右 2 チャンネル分のデータを書き込みます。
  for(var i:int = t; (t<i+8192); ++t) {
    xval = yval = 0;
    for(var pitch_list_num in pitch_list) {
      switch(envelope_mode[pitch_list[pitch_list_num]]){
        case "stop":
        envelope_level[pitch_list[pitch_list_num]] = 0;
        break;

        case "attack":
        if(t%10000==0) trace(pitch_list);
        if(envelope_time[pitch_list[pitch_list_num]]< attack_time){
          envelope_level[pitch_list[pitch_list_num]] =
           envelope_time[pitch_list[pitch_list_num]]/attack_time;
          envelope_tmp[pitch_list[pitch_list_num]] = envelope_level[pitch_list[pitch_list_num]]; //Attack
中に Release した時用
          envelope_time[pitch_list[pitch_list_num]] += 1/44100;
        } else {
          envelope_mode[pitch_list[pitch_list_num]] = "decay";
          envelope_time[pitch_list[pitch_list_num]] = 0;
        }
        break;

        case "decay":
        if(envelope_time[pitch_list[pitch_list_num]]< decay_time){
          envelope_level[pitch_list[pitch_list_num]] =
           1-(1-sustain_volume)*envelope_time[pitch_list[pitch_list_num]]/decay_time;
          envelope_tmp[pitch_list[pitch_list_num]] = envelope_level[pitch_list[pitch_list_num]]; //Decay
中に Release した時用
          envelope_time[pitch_list[pitch_list_num]] += 1/44100;
        } else {
          envelope_mode[pitch_list[pitch_list_num]] = "sustain";
          envelope_time[pitch_list[pitch_list_num]] = 0;
        }
        break;

        case "sustain":
        break;

        case "release":
        if(envelope_time[pitch_list[pitch_list_num]]< release_time){
          envelope_level[pitch_list[pitch_list_num]] =
```

```
           envelope_tmp[pitch_list[pitch_list_num]]-
           envelope_tmp[pitch_list[pitch_list_num]]*
           envelope_time[pitch_list[pitch_list_num]]/release_time;
          envelope_time[pitch_list[pitch_list_num]] += 1/44100;
        } else {
          envelope_mode[pitch_list[pitch_list_num]] = "stop";
          envelope_time[pitch_list[pitch_list_num]] = 0;
          pitch_list = except(pitch_list, pitch_list[pitch_list_num]);
        }
        break;
      }
      freq = pitchToFreq(pitch_list[pitch_list_num]);
      xval += velocity[pitch_list[pitch_list_num]] / 100
 * envelope_level[pitch_list[pitch_list_num]] * ptar[int(freq*t*n/44100)%n].x/50;
      yval += velocity[pitch_list[pitch_list_num]] / 100
 * envelope_level[pitch_list[pitch_list_num]] * ptar[int(freq*t*n/44100)%n].y/50;
    }
    // WaveSpectra に合わせてスケール変更@ analytic_signal_synthesizer10.fla
    e.data.writeFloat(-yval/2);
    e.data.writeFloat(xval/2);
  }
}

// Max/MSP との接続
var socket :XMLSocket = new XMLSocket();
socket.connect( "127.0.0.1", 31337 );
socket.addEventListener( Event.CONNECT, connectHandler );
socket.addEventListener( DataEvent.DATA, dataHandler );
function connectHandler ( evt :Event ) :void   // 接続時イベント
{
  trace( "connect" );
}
function dataHandler ( evt :DataEvent ) :void   // 受信時イベント
{
  var pitchAndVelocity = new Array(2);
  var pitch;
  pitchAndVelocity = evt.data.split(";"); //最後のセミコロンを取り除くための split
  pitchAndVelocity = pitchAndVelocity[0].split(" "); //セミコロン以外をスペースで split
  pitch = pitchAndVelocity[0];
  if(pitchAndVelocity[1] == 0) {
    envelope_mode[pitch] = "release";
    envelope_time[pitch] = 0;
    if(pitch >= key_min && pitch <= key_max)   note[pitch].prevFrame();
  } else {
    velocity[pitch] = pitchAndVelocity[1];
    envelope_mode[pitch] = "attack";
    envelope_time[pitch] = 0;
    pitch_list.push(pitch);
    if(pitch >= key_min && pitch <= key_max)   note[pitch].nextFrame();
  }
}

var pitch_min = 36, pitch_max = 84;
var key_min=48, key_max=83;
var pitch_list = new Array();
var velocity = new Array();
var envelope_time = new Array();
var envelope_level = new Array();
```

```
var envelope_tmp = new Array();
var envelope_mode = new Array();
var key_flag = new Array();
for(i=pitch_min; i<=pitch_max; i++) {
  velocity[i] = 0;
  envelope_time[i] = 0;
  envelope_level[i] = 0;
  envelope_tmp[i] = 0;
  envelope_mode[i] = "stop";
  key_flag[i] = false; //キーボード押しっぱなし対策
}
var mouse_flag:Boolean = false;
var note = new Array();
note[48] = C3_mc;
note[49] = CS3_mc;
note[50] = D3_mc;
note[51] = DS3_mc;
note[52] = E3_mc;
note[53] = F3_mc;
note[54] = FS3_mc;
note[55] = G3_mc;
note[56] = GS3_mc;
note[57] = A3_mc;
note[58] = AS3_mc;
note[59] = B3_mc;
note[60] = C4_mc;
note[61] = CS4_mc;
note[62] = D4_mc;
note[63] = DS4_mc;
note[64] = E4_mc;
note[65] = F4_mc;
note[66] = FS4_mc;
note[67] = G4_mc;
note[68] = GS4_mc;
note[69] = A4_mc;
note[70] = AS4_mc;
note[71] = B4_mc;
note[72] = C5_mc;
note[73] = CS5_mc;
note[74] = D5_mc;
note[75] = DS5_mc;
note[76] = E5_mc;
note[77] = F5_mc;
note[78] = FS5_mc;
note[79] = G5_mc;
note[80] = GS5_mc;
note[81] = A5_mc;
note[82] = AS5_mc;
note[83] = B5_mc;
var key = new Array();
key[48] = "z";
key[49] = "s";
key[50] = "x";
key[51] = "d";
key[52] = "c";
key[53] = "v";
key[54] = "g";
key[55] = "b";
```

```
key[56] = "h";
key[57] = "n";
key[58] = "j";
key[59] = "m";
key[60] = ",";
key[61] = "l";
key[62] = ".";
key[63] = ";"
key[64] = "/";
key[65] = "\\";
key[66] = "]";
key[67] = "q";
key[68] = "2";
key[69] = "w";
key[70] = "3";
key[71] = "e";
key[72] = "r";
key[73] = "5";
key[74] = "t";
key[75] = "6";
key[76] = "y";
key[77] = "u";
key[78] = "8";
key[79] = "i";
key[80] = "9";
key[81] = "o";
key[82] = "0";
key[83] = "p";

this.stage.addEventListener(KeyboardEvent.KEY_DOWN, key_down);
function key_down(event:KeyboardEvent){
  var key_str:String;
  for(var key_num=key_min; key_num<key_max; key_num++) {
    if(event.keyCode == 188) key_str = ",";
    else if(event.keyCode == 190) key_str = ".";
    else if(event.keyCode == 187) key_str = ";";
    else if(event.keyCode == 191) key_str = "/";
    else if(event.keyCode == 226) key_str = "\\";
    else key_str = String.fromCharCode(event.keyCode);
    if(key_str.toLowerCase() == key[key_num]) {
      if(key_flag[key_num] == false) {
        envelope_mode[key_num] = "attack";
        envelope_time[key_num] = 0;
        velocity[key_num] = 100;
        pitch_list.push(key_num);
        note[key_num].nextFrame();
      }
      key_flag[key_num] = true;
    }
  }
};
for(var key_num=key_min; key_num<=key_max; key_num++) {
  note[key_num].addEventListener(MouseEvent.MOUSE_DOWN, mouse_down);
}
function mouse_down(event:MouseEvent){
  for(var key_num=key_min; key_num<key_max; key_num++) {
    if(event.target.name == note[key_num].name) {
      envelope_mode[key_num] = "attack";
```

```
      envelope_time[key_num] = 0;
      velocity[key_num] = 100;
      pitch_list.push(key_num);
      note[key_num].nextFrame();
    }
  }
  mouse_flag = true;
};
for(key_num=key_min; key_num<key_max; key_num++) {
  note[key_num].addEventListener(MouseEvent.MOUSE_OVER, mouse_over);
}
function mouse_over(event:MouseEvent){
  if(mouse_flag == true){
    for(var key_num=key_min; key_num<key_max; key_num++) {
      if(event.target.name == note[key_num].name) {
        envelope_mode[key_num] = "attack";
        envelope_time[key_num] = 0;
        velocity[key_num] = 100;
        pitch_list.push(key_num);
        note[key_num].nextFrame();
      }
    }
  }
};
this.stage.addEventListener(KeyboardEvent.KEY_UP, key_up);
function key_up(event:KeyboardEvent){
  //trace_txt.text = String(event.keyCode);
  var key_str:String;
  for(var key_num=key_min; key_num<key_max; key_num++) {
    if(event.keyCode == 188) key_str = ",";
    else if(event.keyCode == 190) key_str = ".";
    else if(event.keyCode == 187) key_str = ";";
    else if(event.keyCode == 191) key_str = "/";
    else if(event.keyCode == 226) key_str = "\\";
    else key_str = String.fromCharCode(event.keyCode);
    if(key_str.toLowerCase() == key[key_num]) {
      envelope_mode[key_num] = "release";
      envelope_time[key_num] = 0;
      velocity[key_num] = 100;
      key_flag[key_num] = false;
      note[key_num].prevFrame();
    }
  }
};
for(key_num=key_min; key_num<key_max; key_num++) {
  note[key_num].addEventListener(MouseEvent.MOUSE_UP, mouse_up);
}
function mouse_up(event:MouseEvent){
  for(var key_num=key_min; key_num<key_max; key_num++) {
    if(event.target.name == note[key_num].name) {
      envelope_mode[key_num] = "release";
      envelope_time[key_num] = 0;
      velocity[key_num] = 100;
      note[key_num].prevFrame();
    }
  }
  mouse_flag = false;
};
```

```
for(key_num=key_min; key_num<key_max; key_num++) {
  note[key_num].addEventListener(MouseEvent.MOUSE_OUT, mouse_out);
}
function mouse_out(event:MouseEvent){
  if(mouse_flag == true){
    for(var key_num=key_min; key_num<key_max; key_num++) {
      if(event.target.name == note[key_num].name) {
        envelope_mode[key_num] = "release";
        envelope_time[key_num] = 0;
        velocity[key_num] = 100;
        note[key_num].prevFrame();
      }
    }
  }
};

function pitchToFreq(notenum:Number) {
  return Math.floor(440*Math.pow(2, (notenum-69)/12));
};


//エンベロープのグラフ表示
var attack_time, attack_volume, decay_time, sustain_volume, release_time;
function envelope_graph_draw(event:Event) {
  onEnvelopeChanged();
};
function onEnvelopeChanged(): void {
  //envelope_mc.scaleX = 30;
  //envelope_mc.scaleY = -50;
  envelope_mc.graphics.clear();
  envelope_mc.graphics.lineStyle(0, 0x000000);
  envelope_mc.graphics.moveTo(0,0);
  envelope_mc.graphics.lineTo(attack_mc.x-envelope_mc.x, attack_mc.y-envelope_mc.y);
  envelope_mc.graphics.lineTo(sustain_mc.x-envelope_mc.x, sustain_mc.y-envelope_mc.y);
  envelope_mc.graphics.lineTo(sustain_mc.x+sustain_mc.width-envelope_mc.x, sustain_mc.y-envelope_mc.y);
  envelope_mc.graphics.lineTo(release_mc.x-envelope_mc.x, 0);
  attack_time = (attack_mc.x-envelope_mc.x)/100
  attack_volume = (envelope_mc.y-attack_mc.y)/100;
  decay_time = (sustain_mc.x-attack_mc.x)/100;
  sustain_volume = (envelope_mc.y-sustain_mc.y)/100;
  release_time = (release_mc.x-sustain_mc.x-sustain_mc.width)/100;
};
onEnvelopeChanged();

attack_mc.addEventListener(MouseEvent.MOUSE_UP, envelope_graph_draw);
sustain_mc.addEventListener(MouseEvent.MOUSE_UP, envelope_graph_draw);
release_mc.addEventListener(MouseEvent.MOUSE_UP, envelope_graph_draw);
this.stage.addEventListener(Event.ENTER_FRAME, envelope_graph_draw);

//エンベロープのドラッグ設定
attack_mc.addEventListener(MouseEvent.MOUSE_DOWN, attack_startDrag);
function attack_startDrag(event:Event) {
  attack_mc.startDrag(false, new Rectangle(envelope_mc.x, envelope_mc.y-100,
 sustain_mc.x-envelope_mc.x, 100+sustain_mc.y-envelope_mc.y));
};
sustain_mc.addEventListener(MouseEvent.MOUSE_DOWN, sustain_startDrag);
function sustain_startDrag(event:Event) {
  sustain_mc.startDrag(false, new Rectangle(attack_mc.x, attack_mc.y,
```

```
   release_mc.x-attack_mc.x-sustain_mc.width, release_mc.y-attack_mc.y));
};
release_mc.addEventListener(MouseEvent.MOUSE_DOWN, release_startDrag);
function release_startDrag(event:Event) {
  release_mc.startDrag(false, new Rectangle(sustain_mc.x+sustain_mc.width, envelope_mc.y, 100, 0));
};
this.stage.addEventListener(MouseEvent.MOUSE_UP, envelope_stopDrag);
function envelope_stopDrag(event:Event) {
  attack_mc.stopDrag();
  sustain_mc.stopDrag();
  release_mc.stopDrag();
};

var pulse_ar:Array=new Array(n);

var ptar:Array=new Array(n);
var oldptar:Array=new Array(n);
var locator:Sprite=new Sprite();
locator.x=z_mc.x;
locator.y=z_mc.y;
var oldLocator:Sprite=new Sprite();
oldLocator.x=locator.x;
oldLocator.y=locator.y;
addChild(locator);
for(i=0; i<n; i++){
  ptar[i] = new Point();
  ptar[i].y=Math.cos(2*Math.PI*i/n)*50;
  ptar[i].x=Math.sin(2*Math.PI*i/n)*50;
  ptar[i].addEventListener(MouseEvent.MOUSE_DOWN, updt);
  ptar[i].addEventListener(MouseEvent.MOUSE_UP, stp);
  ptar[i].ind=i;
  locator.addChild(ptar[i]);

  oldptar[i] = new Point();
  oldptar[i].x=ptar[i].x;
  oldptar[i].y=ptar[i].y;
//  ptar[i].ind=i;
  oldLocator.addChild(oldptar[i]);
}


addEventListener(Event.ENTER_FRAME, mv);
yListPlot(ptar);
zListPlot(ptar);
ASSoundPlay();
var trig:int = 0;
var pos:int;
function updt(event:MouseEvent) {
  trig = 1;
  pos = event.target.ind;
  for(var i=0; i<n; i++){
    oldptar[i].x = ptar[i].x;
    oldptar[i].y = ptar[i].y;
  }
}
function mv(event:Event) {
  if(trig==1){
    //ptar[pos].x=oldx+(mouseX-oldx)*1-(mouseY-oldy)*0;
```

```
    //ptar[pos].y=oldy+(mouseX-oldx)*0+(mouseY-oldy)*1;
    for(var i=pos; i<64+pos; i++){
      var dy = locator.mouseY-oldptar[pos].y;
      var dx = locator.mouseX-oldptar[pos].x;
      var z = mult([dy,dx],[pulsereal_array[(-i+n+pos)%n],pulseimag_array[(-i+n+pos)%n]])
      ptar[i%n].y = oldptar[i%n].y + z[0];
      ptar[i%n].x = oldptar[i%n].x + z[1];
    }
  }
  yListPlot(ptar);
  zListPlot(ptar);
}
function stp(event:MouseEvent) {
  trig = 0;
}

//楽器の読み込み
var zero = new Array([],[]);
var sine = new Array([],[]);
for(i=0; i<n; i++){
  zero[0][i]=0;
  zero[1][i]=0;
  sine[0][i]=Math.sin(2*Math.PI*i/n);
  sine[1][i]=Math.cos(2*Math.PI*i/n);
}
instrument_ComboBox.addItem({label:"zero", data:zero});
instrument_ComboBox.addItem({label:"sine", data:sine});
instrument_ComboBox.selectedIndex = 1; //デフォルトで"sine"を選択
instrument_ComboBox.addItem({label:"Grand Piano", data:[[0.130737305,0.019104004,-0.092987061,-0.194641113,-
],[-0.671936035,-0.663909912,-0.637390137,-0.580474854,-0.499084473,-0.409576416,-0.324188232,-0.246856689,-
]]});
instrument_ComboBox.addItem({label:"El.Piano", data:[[-0.623077393,-0.587738037,-0.547332764,-0.501708984,-0
],[0.265808105,0.339630127,0.399108887,0.459289551,0.507202148,0.542114258,0.57635498,0.608764648,0.6359558
]]});
instrument_ComboBox.addItem({label:"Church Organ", data:[[0.415496826,0.197875977,-0.037384033,-0.130706787,
],[-0.440704346,-0.499603271,-0.415496826,-0.202514648,-0.037231445,0.034301758,0.046295166,-0.026855469,-0.
]]});
instrument_ComboBox.addItem({label:"Harmonica", data:[[0.106170654,0.209014893,0.357727051,0.484527588,0.557
],[0.474121094,0.492248535,0.469390869,0.375091553,0.275634766,0.191650391,0.049591064,-0.202453613,-0.49713
]]});
instrument_ComboBox.addItem({label:"Nylon Guiter", data:[[-0.319458008,-0.222412109,-0.100280762,0.011627197
],[0.58883667,0.654266357,0.688201904,0.683197021,0.670654297,0.629089355,0.566192627,0.500762939,0.42279052
]]});
instrument_ComboBox.addItem({label:"Violin", data:[[-0.122070313,-0.125854492,-0.09854126,0.122558594,0.3932
],[0.382629395,0.427368164,0.56137085,0.686401367,0.571685791,0.345031738,0.170471191,0.073181152,0.14593505
]]});
instrument_ComboBox.addItem({label:"Chor Aahs", data:[[0.017150879,0.042572021,0.056396484,0.058349609,0.053
],[0.274108887,0.232727051,0.193054199,0.156219482,0.120391846,0.085693359,0.063354492,0.065765381,0.0897521
]]});
instrument_ComboBox.addItem({label:"Voice Oohs", data:[[0.084533691,0.01184082,-0.056793213,-0.127197266,-0.
],[-0.677368164,-0.684967041,-0.685760498,-0.687225342,-0.678131104,-0.648345947,-0.62008667,-0.591186523,-0
]]});
instrument_ComboBox.addItem({label:"Synth Voice", data:[[0.572723389,0.58190918,0.58807373,0.599090576,0.607
],[0.213043213,0.162261963,0.115814209,0.069671631,0.013000488,-0.045501709,-0.093994141,-0.143920898,-0.191
]]});
instrument_ComboBox.addItem({label:"Trumpet", data:[[0.999969482,0.966827393,0.820373535,0.630187988,0.42697
],[0.113372803,-0.193267822,-0.432647705,-0.585113525,-0.66293335,-0.674346924,-0.638580322,-0.578796387,-0.
]]});
```

```
instrument_ComboBox.addItem({label:"Flute", data:[[-0.263214111,-0.309631348,-0.353881836,-0.405548096,-0.45
],[-0.110656738,-0.083251953,-0.046081543,0.005889893,0.083129883,0.186035156,0.313568115,0.453277588,0.5776
]]});
instrument_ComboBox.addItem({label:"Recorder", data:[[0.704040527,0.687957764,0.67364502,0.648376465,0.61730
],[-0.015014648,-0.080169678,-0.145172119,-0.210632324,-0.270050049,-0.32144165,-0.365509033,-0.400695801,-0
]]});
instrument_ComboBox.addItem({label:"Ocarina", data:[[0.722045898,0.709899902,0.694580078,0.670135498,0.64492
],[-0.014465332,-0.088348389,-0.157745361,-0.223266602,-0.284820557,-0.344055176,-0.402526855,-0.459014893,-
]]});
instrument_ComboBox.addItem({label:"Square Lead", data:[[-0.285797119,-0.276153564,-0.297637939,-0.33245849
],[0.054016113,0.001586914,-0.032714844,-0.045959473,-0.04196167,-0.027191162,-0.010101318,0.018371582,0.087
]]});
instrument_ComboBox.addItem({label:"Saw Lead", data:[[-0.139648438,-0.121520996,-0.1015625,-0.080810547,-0.0
],[0.279449463,0.293457031,0.307800293,0.321380615,0.335540771,0.346679688,0.358673096,0.374084473,0.380981
]]});
instrument_ComboBox.addItem({label:"Echoes", data:[[-0.624237061,-0.574676514,-0.497467041,-0.394470215,-0.2
],[0.182403564,0.308349609,0.407836914,0.480010986,0.513092041,0.526641846,0.530090332,0.511688232,0.499786
]]});
//instrument_ComboBox.addItem({label:"", data:[[],[]]});
instrument_ComboBox.addEventListener(Event.CHANGE, program_change);
instrument_ComboBox.text = "sine";
function program_change(event:Event) {
  //trace(event.target.selectedItem.data);
  for(var i=0; i<n; i++){
    ptar[i].x=event.target.selectedItem.data[0][i]*50;
    ptar[i].y=event.target.selectedItem.data[1][i]*50;
  }
  this.stage.focus = null; //鍵盤を押せるようにコンボボックスのフォーカスを外す
};


// 印加信号制御
pulse_ComboBox.addItem({label:"0 (Move)", data:[pulsereal0_array,pulseimag0_array]});
pulse_ComboBox.addItem({label:"10 (Low freq.)", data:[pulsereal10_array,pulseimag10_array]});
pulse_ComboBox.addItem({label:"20 (Low-middle freq.)", data:[pulsereal20_array,pulseimag20_array]});
pulse_ComboBox.addItem({label:"50 (High-middle freq.)", data:[pulsereal50_array,pulseimag50_array]});
pulse_ComboBox.addItem({label:"100 (High freq.)", data:[pulsereal100_array,pulseimag100_array]});
pulse_ComboBox.addItem({label:"inf. (click pulse)", data:[pulserealinf_array,pulseimaginf_array]});
pulse_ComboBox.selectedIndex = 1; //デフォルトで"10 (Low freq.)"を選択
pulse_ComboBox.addEventListener(Event.CHANGE, pulse_change);
function pulse_change(event:Event) {
  //trace(event.target.selectedItem.data);
    pulsereal_array = event.target.selectedItem.data[0];
    pulseimag_array = event.target.selectedItem.data[1];
    this.stage.focus = null; //鍵盤を押せるようにコンボボックスのフォーカスを外す
};

addEventListener(MouseEvent.MOUSE_WHEEL,wheel);
var wheeltmp:int=1;
function wheel(e:MouseEvent){
  wheeltmp += (e.delta>0)?1:-1;
  wheeltmp = (wheeltmp<=0)?0:(wheeltmp>=5)?5:wheeltmp;
  switch(wheeltmp){
    case 0:
    pulsereal_array=pulsereal0_array;
    pulseimag_array=pulseimag0_array;
    break;
    case 1:
```

```
    pulsereal_array=pulsereal10_array;
    pulseimag_array=pulseimag10_array;
    break;
    case 2:
    pulsereal_array=pulsereal20_array;
    pulseimag_array=pulseimag20_array;
    break;
    case 3:
    pulsereal_array=pulsereal50_array;
    pulseimag_array=pulseimag50_array;
    break;
    case 4:
    pulsereal_array=pulsereal100_array;
    pulseimag_array=pulseimag100_array;
    break;
    case 5:
    pulsereal_array=pulserealinf_array;
    pulseimag_array=pulseimaginf_array;
    break;
  }
  pulse_ComboBox.selectedIndex = wheeltmp;
}

function yListPlot(li:Array):void {
  ygraph.graphics.clear();
  ygraph.graphics.lineStyle(0, 0x6600CC);
  ygraph.graphics.moveTo(0, li[0].y);

  //2 周期分プロット
  for (var i:int = 1; i <= li.length*2; i++) {
    ygraph.graphics.lineTo(200*i/n/2, li[i%n].y);
  }
}
function zListPlot(li:Array):void {
  zgraph.graphics.clear();
  zgraph.graphics.lineStyle(0, 0x000000);
  zgraph.graphics.moveTo(li[0].x, li[0].y);

  for (var i:int = 1; i <= li.length; i++) {
    zgraph.graphics.lineTo(li[i%n].x, li[i%n].y);
  }
}

function except(li:Array, val:Number) {
  var li2 = new Array();
  for(var i in li) {
    if(li[i] != val) li2.push(li[i]);
  }
  return li2;
};
```

参考文献

[1] 岩淵勇樹, 秋田純一, 北川章夫. 閉曲線図形に基づいた音色生成方法の検討. エンタテインメントコンピューティング 2008 論文集, pp. 143–146, 2008.

[2] 岩淵勇樹, 秋田純一, 北川章夫. 閉曲線図形の特性に基づいた音色生成の一手法. 第 16 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2008) 論文集, pp. 149–150, 2008.

[3] 青柳龍也, 小坂直敏, 平田圭二, 堀内靖雄. コンピュータ音楽—歴史・テクノロジー・アート—. 東京電機大学出版局, 2001.

[4] 小坂直敏. コンピュータ音楽で用いる珍しい音合成方式. 音響学会音楽音響研究会資料, 2005.

[5] R. Boulanger, P. Smaragdis, and J. Ffitch. Scanned synthesis: An introduction and demonstration of a new synthesis and signal processing technique. In *Proceedings of the 2000 International Computer Music Conference*, pp. 372–375, 2000.

[6] cSounds.com. scanned synthesis. http://www.csounds.com/scanned/.

[7] KORG. Kaossilator. http://www.korg.co.jp/Product/Dance/kaossilator/.

[8] U&I Software. Metasynth 5. http://www.uisoftware.com/MetaSynth/.

[9] J. Niinisalo. The aphex face. http://www.bastwood.com/aphex.php.

[10] S. Malinowski. The music animation machine. http://www.musanim.com/.

[11] Y.Nishibori and T.Iwai. Tenori-on. In *Proceedings of 6th International Conference on New Interfaces for*, pp. 172–175, 2006.

[12] Shou. Swave. http://sky.geocities.jp/izeefss/sakurayamaato/.

[13] W.S. Yeo and J. Berger. Application of raster scanning method to image sonification, sound visualization, sound analysis and synthesis. In *Proc of the 9th Int. Conference on Digital Audio Effects. Montreal, Canada*, 2006.

[14] G.Levin and Z.Lieberman. Sounds from shapes: Audiovisual performance with hand silhouette contours in the manual input sessions. In *Proceedings of 5th International Conference on New Interfaces for Musical Expression*, pp. 26–28, 2005.

[15] 長嶋洋一. 新しい筋電楽器のための筋電情報認識手法. 第 85 回音楽情報科学研究会, Vol. 2010, No. 1, pp. 1–6, 2010.

[16] D. Rokeby. Very nervous system. http://homepage.mac.com/davidrokeby/vns.html.

[17] A.Camurri, M.Ricchetti, and R.Trocca. Eyesweb–toward gesture and affect recognition in dance/music interactive systems. In *IEEE International Conference on Multimedia Computing and Systems*, Vol. 1, pp. 643–648, 1999.

[18] M.Funk, K.Kuwabara, and M.J.Lyons. Sonification of facial actions for musical expression. In *Proceedings of NIME '03*, pp. 127–131, 2003.

[19] M.J.Lyons. Facial gesture interfaces for expression and communication. In *Proceedings IEEE Inteernational Conference on Systems, Man and Cybernetics*, Vol. 1, pp. 598–603, 2004.

[20] M.J.Lyons, M.Haehnel, and N.Tetsutani. Designing, playing, and performing with a vision-based mouth interface. In *Proceedings of NIME '03*, pp. 116–121, 2003.

[21] 岩淵勇樹, 秋田純一, 北川章夫. 閉曲線を利用した音色操作方法の検討と実装. 第 85 回音楽情報科学研究会, Vol. 2010, No. 12, pp. 1–3, 2010.