音楽情報処理の技術的基盤

平成 4 年度文部省科学研究費総合研究 (B)

「音楽情報科学に関する総合的研究」

課題番号 **04352030**

調査報告

平成 5 年 3 月

目次

はじめに **1**

1 音響信号処理 **3**
1.1 楽音分析技術 (柏野) . . . 3
1.1.1 楽音分析研究の展開 . . . 3
1.1.2 短時間フーリエ解析 . . . 4
1.1.3 帯域フィルタバンク . . . 6
1.1.4 ウェーブレット解析 . . . 7
1.1.5 ウィグナー分布 . . . 8
1.1.6 その他の方法 . . . 10
1.1.7 まとめ . . . 12
1.2 楽音合成技術 (長嶋) . . . 12
1.2.1 楽音合成の要素技術 . . . 12
1.2.2 加算型 . . . 15
1.2.3 減算型 . . . 17
1.2.4 非線形変換 . . . 19
1.2.5 サンプリング . . . 20
1.2.6 グラニュラーシンセシス . . . 21
1.2.7 楽音合成技術の展望 . . . 22
1.3 楽音と音場のシミュレーション技術 (藤森) . . . 23
1.3.1 ワールド・モデル – 楽器、演奏者、音場、聴取者 . . . 23
1.3.2 楽器 – 演奏者系 . . . 24
1.3.3 室内音場 – 聴取者系 . . . 35
参考文献 . . . 36

2 音楽情報の表現 **45**
2.1 楽譜表記方式 (松島) . . . 45
2.2 音楽情報の表現スキーム (平田) . . . 49
2.2.1 Lisp . . . 50
2.2.2 オブジェクト指向的表現 . . . 52
2.2.3 制約 . . . 60
2.2.4 ニューラルネットワーク . . . 65
2.2.5 CSP – Communicating Sequential Processes . . . 70
2.2.6 論理型プログラミング言語 . . . 75
2.3 標準化について (長嶋, 松島) . . . 75
参考文献 . . . 78

3 音楽認識 **83**
3.1 楽譜認識 (松島) 83
3.1.1 楽譜認識の問題点 85
3.1.2 画像の読み込み 86
3.1.3 五線の検出 87
3.1.4 記号の認識 87
3.1.5 後処理 91
3.1.6 市販システム 92
3.2 採譜 (片寄) 92
3.2.1 楽音への群化 92
3.2.2 音楽分析 97
3.2.3 楽譜化 101
3.3 演奏表現の生成 (片寄) 101
3.4 感性情報の処理 (片寄) 104
参考文献 108

4 インタラクティブシステム **118**
4.1 新世代楽器 (鈴木) 118
4.1.1 RADIO DRUM 118
4.1.2 MIDI BATON 120
4.1.3 VIDEO HARP 122
4.1.4 HYPERINSTRUMENT 127
4.1.5 まとめ 130
4.2 音楽制作ツール 131
4.2.1 UPIC (鈴木) 131
4.2.2 MAX (平野) 135
4.2.3 ISPW (平野) 144
4.2.4 その他のシステム (鈴木) 148
4.3 伴奏システム (堀内) 150
4.3.1 伴奏システム研究の流れ 150
4.3.2 カーネギー・メロン大学での研究 151
4.3.3 マサチューセッツ工科大学メディア・ラボでの研究 162
4.3.4 コネティカット大学での研究 165
4.3.5 クィーンズ大学での研究 167
4.3.6 早稲田大学（大照研究室）での研究 168
4.3.7 大阪大学（井口研究室）での研究 169
4.3.8 東京工業大学（田中研究室）での研究 169
4.3.9 おわりに 170
参考文献 170

5 音楽情報科学に関する諸分野 **174**
5.1 コンピュータ音楽の制作環境 (嶋津) 174
5.1.1 音楽とテクノロジーの距離 174
5.1.2 コンピュータ音楽の歴史的概要 174
5.1.3 わが国における展開 178

5.1.4 作曲環境上の課題 181
5.1.5 コンピュータ音楽の意義 182
5.2 音楽学における計算機応用 (鈴木) 183
5.3 音楽と脳波信号処理 (才脇) 186
5.3.1 音楽と脳 186
5.3.2 音楽傾聴時の脳波解析 188
5.3.3 PET と MRI を用いた研究例 [b39] 193
5.3.4 番外編: ICMC にも登場した脳波 !? 194

本調査報告は、平成 4 年度文部省科学研究費総合研究 (B)「音楽情報科学に関する総合的研究」(下注) の研究活動の一環として、計算機音楽と音楽情報科学に関わる諸研究を技術的視点から調査し、その現状を解説したものである。

この作業は、研究分担者の中の 5 人と、このために参加いただいた 6 人の研究協力者により共同して進められた。これらの方々はいずれも本分野で活発に研究活動、創作活動にたずさわっており、それぞれが専門とする項目を分担して執筆した。ここにまとめて執筆者名と所属を記す。

柏野 邦夫　東京大学大学院 工学系研究科 情報工学専門課程
片寄 晴弘　(財) イメージ情報科学研究所
才脇 直樹　大阪大学 基礎工学部 システム工学科
嶋津 武仁　福島大学 教育学部
鈴木 孝　東京工業高等専門学校
長嶋 洋一　Art & Science Laboratory
平田 圭二　(財) 新世代コンピュータ技術開発機構
平野 砂峰旅　Hybridoma Project
藤森 潤一　ヤマハ株式会社 電子楽器事業本部
堀内 靖雄　東京工業大学 理工学研究科 情報工学専攻
松島 俊明　東邦大学 理学部

研究代表者 田口 友康
甲南大学 理学部

(注) 課題番号 04352030。本課題の研究グル - プは 1993 年 9 月に早稲田大学において開催される国際研究集会 ICMC1993 (コンピュータ音楽国際会議、International Computer Music Conference 1993) を研究内容面から企画準備する目的をもって組織された。

はじめに

音楽には、オルガンやピアノなどの楽器製作に端的に見られるように、他の芸術に較べて科学技術とのかかわり合いに本質的に依存する部分が多いように思われる。歴史をたどってみても、17 世紀に遡るオルゴールや演奏装置の製作、産業革命後の 19 世紀後半から 20 世紀初頭に隆盛した各種の自動楽器製作等を経て、電気音響技術から近年の計算機とディジタル信号処理技術まで、つねに時代の新技術をそのなかに取り込んできた。

本報告は、それらのうち近年急速に研究が進展しつつある主要技術と関連分野、とりわけ計算機とディジタル信号処理技術に関わる研究の現状を、第 1 章 音響信号処理、第 2 章 音楽情報の表現、第 3 章 音楽認識、第 4 章 インタラクティブシステム、第 5 章 音楽情報科学に関する諸分野、の 5 章にわけて調査、執筆したものである。

ところで、ここ数年来情報科学の分野において感性情報という分野が認知されはじめて来た。実際に感性情報科学がどのような研究分野であるのかは、これから個々の研究対象と研究方法、およびその成果の積み重ねによって示されて行くことになろうが、一般的には、

情報伝達における主体者としての受け手
受け手による解釈の多義性の積極的な容認
時間発展性、空間性

がそれらの共通的な特徴づけになると筆者は考えている。従来の情報科学が主な対象にしてきた文字、数値情報の生成、伝送、処理、蓄積の技術においては、情報生成・伝達行為の主体者は基本的に送り手だけであり、そこで扱われる情報は、つねに記号化されており、いつでもどこでも、提示の順序によらず、正しいのでなければ誤りであり、受け手の違いに依存せず、解釈の一義性が前提とされるような、いわば「硬い」情報であって、上記のような視点からの取り扱いはなかったか、少なくとも研究の主題ではなかったといえる。

音楽情報科学がこの視点から研究されるとすれば必然的に、音楽学、心理学、認知科学、音響学、生理学等の諸科学との学際的領域に分け入ることになる。本報告でも第 4 章および第 6 章でそれらのいくつかの主題を取り扱っている。とはいえ、特定の執筆者達が限られた時間の中でそのような学際領域にわたる研究のすべてを網羅することは不可能であり、またその必要も無いと考え、主として技術的視点から諸研究の現状を調査してこの報告をまとめた。そのため、例えば第 4 章の主題である音楽認識では、音楽の「機械認識」関連の研究に重点が置かれたものになっている。

計算機と音楽に関する研究を解説、展望したわが国の文献には [1,2,3,4] が挙げられるが、それらの文献が刊行されてからすでに相当の時間が経過している。このたびの報告では特に最近の研究の動向に重きを置くと共に、計算機音楽に対しての音楽家の視点からの歴史的展望と現今の課題を論じた節 (第5.1節) を含めるなど、出来るかぎり特色が出るように努めた。本分野の研究の現状を見通すための資料となれば幸いである。

参考文献

[1] 日本音響学会誌 電気音響特集, 37 巻 11 号, 1981.
[2] 日本音響学会誌 小特集 – 自動演奏 –, 41 巻 6 号, 1985.
[3] ビット誌別冊「コンピュ - タと音楽」, 1987 年 9 月.
[4] 情報処理 特集: 計算機と音楽, 29 巻 6 号, 1988.

第 1 章

音響信号処理

1.1　楽音分析技術 (柏野)

1.1.1　楽音分析研究の展開

楽音分析とは、楽器音を分析してその性質を明らかにすることである。楽音と言えば、「ピッチを知覚する音」など各種の定義があるが、本節では、打楽器音なども含め、一般の楽器の音という意味で用いる。

楽音分析研究では、1960 年代前半まではアナログ的分析手法が用いられていた。1962 年、Fletcher が、ソナグラムとアナログ計測器を用いてピアノ、パイプオルガン、弦楽器の倍音ごとの時間変動を記録し、楽器の音質を規定するパラメータの抽出を試みた研究 [FBS62] などはその代表的な例である。1965 年、Cooley と Tukey による 高速フーリエ変換 (FFT : Fast Fourier Transform) アルゴリズムの発表を受けて、ディジタル的手法による楽音分析の研究が急速に進展した。1967 年の Clark らによるオーケストラ楽器音の分析と合成、Freedman による楽器音の立ち上がり部分の分析などがその発端である。現在、管楽器、弦楽器、打楽器、和楽器など多くの種類の楽器に対し、なお活発な研究活動が展開されている。

これらの楽音分析研究の目的は研究者により多様であるが、次の 2 つに大別することが可能であろう。

1. 楽器音の音響信号の特徴と楽器の物理的性質との関連を明らかにする
2. 楽器音の音響信号の特徴と心理物理量との関連を明らかにする

前者は、振動学的または音響学的に、楽器音の発音形態もしくは音響波形としての物理学的事象を解明しようとするものである。この中には、楽器製作過程への反映を意図する研究も含まれる。後者は、楽器音の音色あるいは品質を主観的に計量化したり、楽器音の物理的特徴の与える心理的特性を評価しようとするものである。これらの研究を、ここでは伝統的な楽音分析研究と呼ぶことにしよう。

一方近年では、音楽認識、自動採譜、音源分離といった研究テーマに代表されるように、楽器音を対象とした機械認識の研究が盛んに行われるようになってきた。すなわち、

3. 音楽演奏の音響信号を分析し、含まれる楽器音の認識を行う

ことを目的とする研究の展開である。音楽演奏は通常「楽音」の集まりであるから、これも、広い意味では楽音分析技術に立脚した研究と言えるだろう。このような研究の背景としては、従来より音楽研究の分野において計算機で音楽の収集・分析・可視化を行いたいとするニーズがあったことに加え、近年人工知能研究や信号処理技術の進展に基づいて機械認識技術が発展してきた

こと、また MIDI (Music Instrument Digital Interface) の実用化を契機として計算機が音楽活動の現場に深く浸透したために、計算機を音楽活動のための有用なツールとして利用することが強く期待されるようになったことなどが挙げられる。

もうひとつの関連する動向として、最近、「聴覚的環境理解」(Auditory Scene Analysis) [Bre90] という研究分野の重要性が心理学や工学の研究者の間で認識されてきている。人間は、周囲の状況を把握し理解する上で聴覚的情報を極めて巧妙に利用しているが、聴覚的環境理解は、その情報処理過程を解明すること、あるいは工学的に実現することを目標とした研究分野である。いわばコンピュータビジョン研究の聴覚版と言えるものであるが、音楽演奏シーンの理解も、その中で重要な研究テーマのひとつとして位置づけられるのである。

音響信号処理の観点から見れば、伝統的な楽音分析研究では、通例単一楽器の単音を対象とするために、時間窓を用いたフーリエ変換による短時間スペクトル解析が用いられる場合が多い。即ち、時間窓を用いたフーリエ変換を基本として、対象に応じた若干の工夫を加えることによって目的が達せられる場合が多いのである。例えば、山口ら [YAM77] は、対象とする音響信号の基本周波数とほぼ同期した短区間の時間窓を用いて離散フーリエ変換を行う短時間スペクトル分析法を適用して、 13 種の楽器音について各倍音ごとの振幅エンベロープおよびピッチ変動を抽出している。ところが、楽器音を対象とした機械認識のための楽音分析では、通例複数種類の楽器の多重音を対象とすることが要求されるために、このような方法だけでは必ずしも十分な結果が得られるとは限らない。例えば、対象とする音響信号に複数の楽器音が混在していて単一の基本周波数が定まらない場合、文献 [YAM77] の方法はそのままでは適用できない。このような事情が、機械認識のための楽音分析において、時間窓フーリエ変換による基本的な短時間スペクトル解析にとどまらず、各種の方法が検討される所以のひとつであろう。

本節では、このような状況を踏まえ、伝統的な楽音分析研究に加え新たに展開されている、楽器音を対象とした機械認識研究の立場から、楽音分析へ適用可能と考えられる種々の音響信号処理技術について概観する。まず、フーリエ変換による短時間スペクトル解析について見た後、これと特徴を対比しながら、帯域フィルタバンク、ウェーブレット解析、ウィグナー分布、およびその他いくつかの方法による楽音分析認識手法について見ていくことにしよう。

1.1.2 短時間フーリエ解析

区間 $(-\infty, \infty)$ で定義された絶対可積分な連続関数 $f(t)$ について、次式の $F(\omega)$ は、フーリエ変換と呼ばれる。

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \exp(-j\omega t) dt. \tag{1.1}$$

ここで、j は虚数単位である。このとき、逆フーリエ変換は次のようになる。

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) \exp(j\omega t) d\omega. \tag{1.2}$$

フーリエ変換は、よく知られているように関数のスペクトル分解という意味を持っている。これは、三角関数の直交性から理解することができる。また、角周波数 ω の波を表現する関数は $e^{j\omega t}$ であるから、その強度を $F(\omega)$ で表すことにすれば、式 (1.2) はこれらの積をあらゆる周波数について足し合わせたものがもとの関数になるという関係を示している。

式 (1.1)、(1.2) は連続関数についてのものであったが、計算機で信号のスペクトルを求めるときには、離散的な時刻の信号値のみが利用できる。従って、時間と周波数が離散的な場合のフーリエ変換を考える必要がある。離散フーリエ変換 (DFT : Discrete Fourier Transform) は、

N 個の複素数の組 $f(n), (n = 0, \cdots, N\text{–}1)$ から他の N 個の複素数の組 $F(k), (k = 0, \cdots, N\text{–}1)$ への一次変換であって、定数 W_N を

$$W_N = \exp(\frac{-j2\pi}{N}) \tag{1.3}$$

と定義すれば

$$F(k) = \sum_{n=0}^{N-1} f(n) W_N^{kn}, \qquad (k = 0, 1, \cdots, N\text{–}1) \tag{1.4}$$

のように表される。この式を $f(n)$ について解けば、逆離散フーリエ変換

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) W_N^{-kn}, \quad (n = 0, 1, \cdots, N\text{–}1) \tag{1.5}$$

となる。高速フーリエ変換は、式 (1.4) および (1.5) の持つ性質を巧みに利用して複素乗算回数を大幅に削減したものである。Cooley と Tukey のアルゴリズムの場合、データ数 N に対して複素乗算回数は $(\log_2 N - 1) \cdot (N/2)$ であるが、直接計算の場合の N^2 に比べると、$N = 1024$ の場合 約 $1/220$ の回数となっていることが分かる。

短時間スペクトルの計算では、時間軸上で局所的なスペクトルを得るために、N 標本の信号波形データを切り出し、これに対して高速フーリエ変換の計算を行う。データの切り出しは、元の波形に時間窓を乗ずる操作と考えることができるが、この時間窓の性質には注意が必要である。即ち、フーリエ変換の性質より、時間領域でこれらの乗算を行うことは、周波数領域では、信号のスペクトルに窓関数のフーリエ変換を畳み込む、即ち重みつき移動平均を行うことに相当する。このため窓関数としては、周波数分解能が高く、かつスペクトルの洩れが少ないものを用いる必要がある。これらの要求を同時に満たすことは困難であるが、音響信号処理に向いた特性を持つものとしてハミング (Hamming) 窓、ハニング (Hanning) 窓などがよく用いられている。

さて、フーリエ変換はもともと無限に長い時間について定義されたものであった。これを短時間に限った信号波形について計算することにすれば、時間的な局所性に関する情報が得られる分、周波数の情報に関して代償を払うことになる。即ち、短時間スペクトルにおいて、標本化周期 T[s] および分析に用いる標本数 N と、得られるスペクトルの周波数分解能 Δf[Hz] には次の関係があり、分析に用いる区間の長さが短くなれば、周波数分解能が悪化する。

$$\Delta f = \frac{1}{TN}. \tag{1.6}$$

これは、音楽音響信号を対象とする場合には大きな問題となる。例えば、A_1 (110.0 Hz) の音を対象とし、短時間フーリエ解析によって基本周波数の抽出を行うことを考えてみる。隣の音 $G_1\sharp$(103.8 Hz), $A_1\sharp$ (116.5 Hz) と区別するためには 6 Hz 以上の周波数分解能が必要である。従って、フーリエ変換の窓長は 167 ms 以上にしなくてはならない。しかし、曲によっては 1 音の持続時間が 100 ms を切ることもあり、それに対応する時間分解能も必要である。

この問題に対して有効なのは、複素スペクトルの位相成分に着目する方法である。例えば、複素スペクトルを位相成分によって内挿する方法 [HAR83] が開発されており、音楽認識システムなどに有効に応用されている。いま、観測区間 T、観測区間に対する周波数 f、区間中央での位相 φ なる単一正弦波を

$$y(t) = A \exp\left[j \left\{ \frac{2\pi f\,(t - T/2)}{T} + \varphi \right\} \right] \tag{1.7}$$

と表し、フーリエ変換すると、複素スペクトル

$$\begin{aligned} z_m &= \frac{A}{T}\int_0^T \exp\left\{j\left(\frac{2\pi ft}{T}-\pi f+\varphi\right)\right\}\cdot\exp\left(-j\frac{2\pi mt}{T}\right)dt \\ &= \frac{A\sin(\pi f)}{\pi(f-m)}\exp(j\varphi) \end{aligned} \tag{1.8}$$

が得られる。これによれば、z_m は方向 φ の直線上に乗っており、$m<f$ と $m>f$ に応じてその向きを反転する。また、z_m の方向 φ の成分の逆数 w_m は

$$w_m = \frac{\pi(f-m)}{A\sin(\pi f)} \tag{1.9}$$

となり、m に関する 1 次関数となる。この直線と横軸との交点から周波数 f が求められる。この方法によれば、単一の周波数成分を対象とする場合には非常に高精度の周波数同定が可能である。また、電子オルガンの和音など、複数の周波数成分を含む音響信号に応用した場合にも、各音のピッチがそれぞれ誤差 0.2% 以下で同定できたと報告されている。

この他にも、高速フーリエ変換に目的に応じた工夫を行うことは盛んに試みられている。例えば高澤 [TAK92] は、フーリエ変換の窓の性質を差し引くことにより、近接したふたつの周波数成分を分離抽出する方法を提案し、鐘など打楽器音の定常部の分析に適用して好結果を得ている。

1.1.3 帯域フィルタバンク

スペクトログラムを得る方法として、**帯域フィルタバンク** を用いる場合がある。この方法は、人間の聴覚系に関する生理学的・解剖学的・心理学的知見に基づいた聴覚モデルを意図したものと、そうでないものとに大別される。

聴覚モデルを意図しないものは、中心周波数の少しずつ異なった狭帯域のバンドパスフィルタを多数並列におく構成が一般的である。この場合、入力音響信号に対する各フィルタの応答の RMS 値がスペクトルとなる。バンドパスフィルタには、Q 値を高めれば時間応答性が悪くなるという不確定性があるが、フーリエ変換による短時間スペクトルの場合に固定した時間窓長が存在するのに比べ、各フィルタごとの不確定性となるので、時間・周波数分解能の点で有利な場合がある。また、中心周波数の設定や Q 値の設定にも自由度がある。例えば、フーリエ変換においては周波数軸が本来的には線形となるが、バンドパスフィルタでは、聴覚的特性を反映して対数軸上 (またはメル軸上) に等間隔に配置することが可能である。これらの特性のため、楽器音を対象とする分析にも有効に応用されており、例えば柏野らによる音源分離実験システム [KT92] では、1 オクターブあたり 48 個のバンドパスフィルタを配置したものを、Q 値を変えて 2 セット用意し、セット間の出力の平均値の 2 項演算により周波数特性を先鋭にする方法が用いられている。

聴覚モデルを意図した周波数解析方法も、盛んに研究されている。人間の聴覚系は、解剖学的には耳介から鼓膜に至る外耳、鼓膜から 3 つの耳小骨を経由し蝸牛前庭窓に至る中耳、および蝸牛から成る内耳に分類される。外耳と中耳の伝達関数は、例えば Meddis ら [MH91] によって調べられている。また、蝸牛には有毛細胞を有する基底膜があり、これによって大まかな周波数解析が行われるとされているが、このプロセスのモデルとしては、ガンマトーンフィルタ (gammatone filter) が有名である。中心周波数 f_0 の、n 次の ガンマトーンフィルタのインパルス応答は、次式で表される。

$$gt(t) = at^{n-1}\exp(-2\pi bt)\cos(2\pi f_0 t+\varphi), \quad (t\geq 0). \tag{1.10}$$

これは、Boer ら [BJ78] により、生理学的実験結果の近似式として導かれたものであるが、Patterson らによって心理学的実験結果にも符合することが確かめられている。この他にも、入力音響レベルによって Q 値が異なるという人間の聴覚系における特徴的な性質に着目し、これを近似的に実現したフィルタなど、数多くのモデルの提案がある。

これらのモデルは、音声認識システムのフロントエンド等として応用されており、聴覚モデルとしての精度という観点のみならず、認識性能の観点からも評価が行われている。現在までのところ、聴覚的特性の一側面を取り入れることが認識性能の向上に直接結びつくとは限らないが、たとえそうであっても、聴覚モデルの工学的な研究および応用には大きな意義があることは言うまでもない。更なる研究の進展によって、新たな分析パラダイムが生まれる可能性もある。

1.1.4 ウェーブレット解析

ウェーブレット変換は、時間的にも周波数的にも局在した一つの関数を用意し、これにスケール変換およびシフト変換を作用させて得られる関数の組を基底関数とした関数展開である。これを利用した信号の解析が **ウェーブレット解析** (wavelet analysis) と呼ばれる。ウェーブレット解析は、Morlet により、石油探索をするときの人工地震波の解析の道具として最初に導入された。それ以来、物理学者、数学者、工学者達により理論の数学的基礎づけが行われ、また音響信号処理や画像処理等への応用も試みられてきた。

ウェーブレット変換は、基本ウェーブレット関数 (アナライジングウェーブレット: analyzing wavelet) $\Psi(t)$ を用いて定義される。いま、$\Psi(t)$ は、原点のまわりに幅 T で局在し、そのフーリエ変換が正の周波数領域 $[\omega_L, \omega_H]$ の間に局在しており、特に信号の直流成分は 0 であるような関数であるとする。このとき、区間 $(-\infty, \infty)$ で定義された絶対可積分な連続関数 $f(t)$ に関するウェーブレット変換 $S(a,b)$ は、次のように定義される。

$$S(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} \Psi^*\left(\frac{t-b}{a}\right) f(t)\, dt. \tag{1.11}$$

ここで $*$ は複素共役を表す。ウェーブレット変換は、スケールを表すパラメータ a と位置 (時刻) を表すパラメータ b を動かすことにより Ψ から作り出される関数の族と、解析対象の関数 $f(t)$ との内積として定義されている。$\Psi((t-b)/a)$ は時間軸上で $(b \pm aT/2)$ の間、また周波数軸上で $[\omega_L/a, \omega_H/a]$ の間に局在していることに注意すれば、式 (1.11) の操作によって信号 $f(t)$ のこの領域内の成分のみが抽出されてくることが分かる。式 (1.11) はまた、$S(a,b)$ が $|a|^{-1/2}\Psi^*(-t/a)$ をインパルス応答とするようなフィルタの出力として求められると見ることもできる。

ウェーブレット変換からもとの波形への逆変換は、次の式で与えられる。

$$f(t) = \frac{1}{C_\Psi} \iint_{-\infty}^{\infty} \frac{1}{a^2} \cdot \frac{1}{|a|^{1/2}} \Psi\left(\frac{t-b}{a}\right) da\, db. \tag{1.12}$$

ここで、C_Ψ は Ψ によって決まる定数である。また、時間－周波数空間で標本化した離散的ウェーブレット変換と逆変換は、次のように表される。

$$S_{m,n} = \int_{-\infty}^{\infty} \Psi^*_{m,n}(t)\, f(t)\, dt, \tag{1.13}$$

$$f(t) = k_\Psi \sum_m \sum_n S_{m,n} \Psi_{m,n}(t). \tag{1.14}$$

ここで k_Ψ は正規化定数である。また $\Psi_{m,n}(t)$ は、スケールパラメータ a と位置パラメータ b を離散化した基本ウェーブレット関数であり、

$$\Psi_{m,n}(t) = a_0^{-m/2}\, \Psi(a_0^{-m} x - n b_0) \tag{1.15}$$

である。

基本ウェーブレット関数は、一般には直交性を持たないが、これを互いに直交するように定めた場合、直交ウェーブレット変換と呼ばれる。直交ウェーブレット基底は、$a_0 = 2$ の場合には、Mallat によって、多重解像度表現と結び付けた導出法が示されている。

ウェーブレット変換の特徴の第一は、基底関数が互いに相似という点である。フーリエ変換基底も互いに相似であるが、基底関数が時間的に局在していないために局所的な情報を得ることが難しい。時間窓をかけたフーリエ変換基底は局在しているが、相似性は持っていない。このため、ウェーブレット変換は、関数の局所的な自己相似性を見るのに適していると考えられる。第二の特徴は、スケール変換により基底関数を生成するために、不確定性関係で許される範囲で時間と周波数の両方の情報を得ることができる点である。特に、単一の窓関数を使った短時間フーリエ解析では時間分解能が周波数によらず一定であるのに対して、ウェーブレット解析では、高い周波数成分については時間分解能が高くなり、低い周波数成分に対しては周波数分解能が高くなるという特徴がある。これは、前項で述べた帯域フィルタバンクと同様の特徴である。実際、中心周波数が異なり Q 値が等しい帯域フィルタのインパルス応答は、時間軸の伸縮について相似であるので、ウェーブレット変換は Q 値が一定の帯域フィルタバンクによる分析を一般化して定式化したものと見ることもできる。

音響信号解析への応用例としては、Kronland-Martinet や上野山らの試みを挙げることができる。Kronland-Martinet [Kro88] は、基本ウェーブレット関数として、ウェーブレット変換を振幅成分と位相成分に分けられるように工夫したものを用い、種々の音響信号の解析を行うとともに、音響信号の合成および加工システムを構成している。上野山ら [UEN92] は、打楽器音の分析および合成システムを実装している。打楽器音は、定常的でないために従来のフーリエ変換による方法の適用が難しい。打楽器音に対しウェーブレット変換による表現が最適なものかどうかについては議論の余地もあるが、文献 [UEN92] は、実際にウェーブレット変換の適用を試みた例として着目されるものである。この他、ウェーブレット変換を聴覚初期過程のモデルとしてとらえた音響分析合成の試み [IRI91] も見られる。

ウェーブレット解析では、基本ウェーブレット関数にバリエーションがあるため、その選択が重要である。また、物理的意味の理解しやすい解析結果の表現方法および表示方法を工夫することも必要である。

1.1.5 ウィグナー分布

ウィグナー分布 (Wigner distribution) と呼ばれる信号表現は、もとは 1932 年、量子力学の分野で導入されたものである。1970 年代末、光学の分野への応用が成功を収め、時間 – 周波数表現に興味を持つ研究者に注目されるところとなった。以後、1 次元信号の処理や画像処理への応用が試みられている [Wec90]。

ウィグナー分布の定義を述べるため、解析対象となる 1 次元関数 $f(t)$ について、まずアンビギュイティ関数[1] (ambiguity function) と呼ばれる関数 A_f を次のように定める。

$$A_f(\xi,\alpha) = \int_{-\infty}^{\infty} f\Big(t+\frac{\alpha}{2}\Big)\, f^*\Big(t-\frac{\alpha}{2}\Big)\, \exp(j\xi t)\, dt. \tag{1.16}$$

このとき、ウィグナー分布 W_f は、次のように定義することができる。

$$W_f(t,\omega) = \frac{1}{2\pi} \iint_{-\infty}^{\infty} A_f(\xi,\alpha)\, \exp\{-j(\xi t+\omega\alpha)\}\, d\xi\, d\alpha. \tag{1.17}$$

[1]アンビギュイティ関数の定義は各種あり、式 (1.16) が唯一のものではない。

即ち、W_f は アンビギュイティ関数の 2 次元フーリエ変換である。これは、次のようにも書くことができる。

$$W_f(t,\omega) = \int_{-\infty}^{\infty} f\Big(t+\frac{\alpha}{2}\Big)\, f^*\Big(t-\frac{\alpha}{2}\Big)\, \exp(-j\omega\alpha)\, d\alpha. \tag{1.18}$$

また、Cohen [Coh66] が一般化した時間 – 周波数表現のクラスは

$$C_f(t,\omega,\phi) = \frac{1}{2\pi} \iiint_{-\infty}^{\infty} \phi(\xi,\alpha)\, f\Big(\rho+\frac{\alpha}{2}\Big)\, f^*\Big(\rho-\frac{\alpha}{2}\Big)\, \exp\{j(\xi\rho-\omega\alpha-\xi t)\}\, d\rho\, d\alpha\, d\xi \tag{1.19}$$

と表されるが、これにおいて核関数 $\phi(\xi,\alpha)$ を常に 1 としたものがウィグナー分布となる。なお、詳細は省略するが、これまでに見た短時間フーリエ変換、ウェーブレット変換も、実は Cohen のクラスに属するメンバーのひとつである。

では、ウィグナー分布の直観的イメージを探ってみよう。特に、ウィグナー分布は、窓をかけたフーリエ変換による短時間スペクトルと、いかなる関係にあるのだろうか。ウィグナー分布の基本的な性質のひとつとして、$f(t)$ についてのウィグナー分布 $W_f(t,\omega)$ を t, ω 軸に射影すると、それぞれ信号のエネルギー $|f(t)|^2$、パワースペクトル $|F(\omega)|^2$ となることを証明することができる。また、

$$f(t) = g(t)\, h(t)\text{のとき} \qquad W_f(t,\omega) = \frac{1}{2\pi}\, W_g(t,\omega) \underset{\omega}{\otimes} W_h(t,\omega) \tag{1.20}$$

なる性質があることも示すことができる。ここで、$\otimes_\omega$ は、ω についての畳み込み演算を表す。これらの性質を用いれば、窓をかけたフーリエ変換によるスペクトルを、次のように式変形することができる。

$$|F_{h,t}(\omega)|^2 = \frac{1}{2\pi} \iint_{-\infty}^{\infty} W_f(t',\omega')\, W_h(t-t',\omega-\omega')\, dt'\, d\omega'. \tag{1.21}$$

ここで、$|F_{h,t}(\omega)|^2$ は、信号 $f(t)$ に窓関数 $h(t)$ をかけて計算したフーリエスペクトルを表す。この式を見ると、信号のウィグナー分布を (窓関数のウィグナー分布で) 平滑化したものが短時間フーリエスペクトルになることが分かる。

また、ウィグナー分布と短時間フーリエスペクトルの関係のもう一つの側面として、クロスウィグナー分布 (cross-Wigner distribution)

$$W_{fg}(t,\omega) = \int_{-\infty}^{\infty} f\Big(t+\frac{\alpha}{2}\Big)\, g^*\Big(t-\frac{\alpha}{2}\Big)\, \exp(-j\omega\alpha)\, d\alpha \tag{1.22}$$

を定義すれば、次の式 (1.23) のように、これと短時間フーリエスペクトルとが、直線位相の違いと変数のスケーリングを除けば等価なものであることを示すこともできる。

$$F_{h,t}(\omega) = \frac{1}{2} \exp\Big(-j\frac{t}{2}\Big)\; W_{fh^*}\Big(\frac{t}{2},\; \frac{\omega}{2}\Big). \tag{1.23}$$

以上は、連続関数に対する記述であったが、ウィグナー分布を計算機上で扱うためには、まずこれを離散化しなければならない。離散化したウィグナー分布は Claasen [CM83] らによって次のように与えられている。

$$W_f(t,\omega) = \frac{T}{\pi} \sum_{k=-\infty}^{\infty} f(t+kT)\, f^*(t-kT)\, \exp(-2j\omega kT). \tag{1.24}$$

ここで T は標本化周期である。折り返しを防ぐために、信号の帯域の上限の周波数の 4 倍以上の周波数で標本化しなくてはならない。さらに、実際の計算では積分区間を有限にした疑似ウィグナー分布 (pseudo Wigner distribution) と呼ばれるものを扱う必要がある。

ウィグナー分布を用いて実際の音響信号の解析を行った例は、文献 [CTD83] などに見ることができる。これによれば、ウィグナー分布による解析は雑音に対して敏感であり、結果の解釈にも注意が必要であるが、音声分析や音声認識の手法として有効に用いることができるという。ウィグナー分布を用いる際に注意すべきなのは、クロス項の発生である。例えば、信号 $f(t)$ が $f_1(t)$ と $f_2(t)$ を成分として持つ場合、即ち

$$f(t) = c_1\, f_1(t) + c_2\, f_2(t), \qquad c_1,\, c_2 : const. \tag{1.25}$$

のとき、$f(t)$ のウィグナー分布を求めると、

$$W_f(t,\omega) = |c_1|^2\, W_{f1}(t,\omega) + |c_2|^2\, W_{f2}(t,\omega) + c_1\, c_2^*\, W_{f1,f2}(t,\omega) + c_2\, c_1^*\, W_{f2,f1}(t,\omega) \tag{1.26}$$

となるが、ここで右辺最後の 2 項がクロス項である。このようなクロス項は、信号が N 個の信号成分からなる場合 $N(N-1)/2$ 個生じる。クロス項は、信号 $f(t)$ が 0 の値を持つ領域において一般に 0 でない値を持つため、ウィグナー分布を用いた時間 – 周波数平面上の信号表現において、信号の性質の直観的な把握が困難となりやすい。

近年では、基本的なウィグナー分布にとどまらず、文献 [NAG91] [Ril87] [CW89] などに見るように、得られる結果が時間 – 周波数表現として好ましい性質を持つように、式 (1.19) の Cohen のクラスにおける核関数を工夫する研究が盛んである。また、認識研究の立場からは、ウィグナー分布を平滑化したものが短時間フーリエスペクトルであるという点に注意すれば、短時間フーリエスペクトルどうしのマッチングにより認識を行う方法は、実は既にぼやけたデータどうしを用いてマッチングを試みていることに相当するとする考え方もある。このような観点からも、今後の研究の成果が期待されよう。

1.1.6 その他の方法

ここでは、従来音声信号処理の分野で用いられている分析方法を中心に、楽音分析への応用の見地から概観する。

音声研究の分野で広く用いられている分析方法に、**線形予測分析** (LPC 分析: Linear Predictive Coding) がある。これは、パラメトリックな分析のひとつであり、音響信号波形について線形予測モデルと呼ばれるモデルを立て、このパラメータを定めることによって音響信号波形やスペクトルの性質を表すものである。

音響信号波形の現在の標本値 x_t と、これに隣接する過去の p 個の標本値との間に、次のような線形結合が成り立つと仮定する。

$$x_t + \alpha_1 x_{t-1} + \cdots + \alpha_p x_{t-p} = \varepsilon_t \tag{1.27}$$

ただし、$\{\varepsilon_t\}$ は互いに無相関な確率変数である。このとき、現時点の標本値 x_t を、過去の標本値から線形予測することができる。即ち、予測値を $\hat{x_t}$ とすれば、

$$\hat{x_t} = -\sum_{i=1}^{p} \alpha_i x_{t-i} \tag{1.28}$$

と表すことができる。このとき、式 (1.27) より、

$$\varepsilon_t = x_t - \hat{x_t} \tag{1.29}$$

となるが、式 (1.27) を $\{\alpha_i\}$ を線形予測係数とする線形予測モデルと呼び、ε_i を線形予測残差と呼ぶ。線形予測分析とは、予測残差の 2 乗平均が最小になるように、式 (1.27) または (1.28) の α_i を調整する方法であり、共分散法と自己相関法の 2 つの解法が提案されている。

では、このような操作によって何が得られるのであろうか。ここで、式 (1.27) の両辺の z 変換を考えれば、α_i は、現在の信号値を出力とする次数 p の全極型フィルタ (これを線形予測フィルタと呼ぶ) の係数に相当していることが分かる。音声を分析対象とする場合、音声生成機構のモデルとして、音声波形を音源インパルスによって励振された声道調音等価フィルタの出力であると考えることができるので、線形予測フィルタの係数を定めることは、声道調音等価フィルタの伝達関数を全極型モデルによってパラメトリックに定めていることに他ならない。これによって、音声の持つ重要な特徴である声道における調音状態を効率的に記述することができるのである。

このような考え方は、楽器音を対象とする場合にも適用可能であろう。即ち、対象に対する適切なモデル化を行うことによって、少数のパラメータで効率的に対象の性質を記述できる方法が見い出されるかも知れない。この際、対象のモデルに適合した分析方法の選択が重要であるのは言うまでもない。

また、音声の分野においては、**ケプストラム分析** という手法が用いられることがある。有声音音声のスペクトルには、声帯振動の高調波に由来する微細構造と、声道における共鳴特性を反映した包絡とが積の形で混在している。ケプストラム分析は、これらを分離抽出することを目的としたものである。

音声生成過程のモデルによれば、有声音音声を $x(t)$ 、周期的な声帯振動を $g(t)$、声道調音等価フィルタのインパルス応答を $h(t)$ としたとき、それぞれのフーリエ変換には次のような関係がある。

$$X(\omega) = H(\omega)\, G(\omega) \tag{1.30}$$

式 (1.30) では、H と G が積の形で存在しているが、両辺の対数をとれば、これは和の形となる。

$$\log|X(\omega)| = \log|H(\omega)| + \log|G(\omega)| \tag{1.31}$$

これを周波数 ω を変数としてフーリエ逆変換したものが、ケプストラム $c(\tau)$ である。

$$c(\tau) = \mathcal{F}^{-1}\log|X(\omega)| = \mathcal{F}^{-1}\log|H(\omega)| + \mathcal{F}^{-1}\log|G(\omega)| \tag{1.32}$$

式 (1.31) 右辺の第 1 項はスペクトル包絡、第 2 項は微細構造に相当するので、両者の逆フーリエ変換には大きな違いがある。そこで、特定の周波数成分を抽出するフィルタリングの操作と同様の操作を $c(\tau)$ について行うことにより、両者を分離抽出することができるのである。

ケプストラム分析も、線形予測分析と同様、音声の生成過程のモデルに基づいた分析法であり、その目的に照らせば、楽器音に対してそのまま適用することには限界がある。しかし、対象のモデル化と有意義な演算操作という意味では、楽音分析を考える上でも大いに参考になるであろう。

楽音分析では、表象的 (representational) な結果の表現、即ち分析結果の明示的な表現が要請される。しかし、広く認識までを含めて考えれば、**ネットワーク** や **確率モデル** の応用も考えられる。音声や楽器音の認識を **ニューラルネットワーク** を用いて行おうとする試みは盛んである。ニューラルネットワークとしては、これまでに種々の型が提案されているが、この目的に使われるのは主に、入力層、中間層 (隠れ層)、出力層からなる多層パーセプトロン型である。この型のニューラルネットワークには、教師信号からの誤差を逆に伝搬するバックプロパゲーション (back-propagation: 誤差逆伝搬法) 学習ができることと、教師信号を与えることによりカテゴリー間の相互抑制が有効に形成できることに特徴がある。また、時系列をリカレントネットワー

クで扱うアプローチや、ボルツマンマシンの応用も試みられている。楽器音の識別の研究[ABE91]など、認識に関するいくつかの実験的検討が行われており、今後の研究の進展が期待される。また、音声認識においては、**隠れマルコフモデル** (HMM : Hidden Marcov Model) と呼ばれる確率モデルを利用する方法が一般的となっているが、近年ではこの隠れマルコフモデルを雑音からの目的音声の分離に用いた例も見られる[VK91]。このような確率モデルの応用も、楽器音の機械認識を考える上でのひとつのヒントとなり得るかも知れない。

1.1.7 まとめ

本節では、楽器音を対象とした機械認識研究の立場から、楽音分析手法について述べた。まず短時間フーリエ解析について見た後、これと特徴を対比しながら、帯域フィルタバンクによる解析、ウェーブレット解析、およびウィグナー分布による時間周波数表現について概観し、その他の関連する分析手法にも言及した。各手法については、その性質に関する直観的な記述を中心に述べるとともに、実際の応用事例を紹介し、その特徴を議論した。楽器音を対象とした機械認識は、多くの階層を持つ大変に困難な課題であって、本節に述べたどの単一の手法によっても容易に達成されるものではない。しかし、基本的な短時間フーリエ解析から一歩進んで、特徴ある分析方法の研究が進んでいるのも事実であり、今後の研究の展開が期待される状況である。なお本節では、数学的に厳密な記述や、計算量の検討等は割愛した。実際の応用においては、計算量の考慮等も重要であることを付記したい。

1.2 楽音合成技術 (長嶋)

本節では、コンピュータ音楽にとって重要な材料である「楽音」を、デジタル信号処理技術によって合成 (シンセサイズ) するための手法に関する研究を紹介する。

まず最初に、基本的な要素「ユニットジェネレータ」を例にとって、楽音合成に関連した基礎概念を確認し、次いで一般的な楽音合成方式の分類に従って、それぞれ具体的な応用研究例の紹介とともに概観する。

なお、本節では原材料となる楽音の合成方式について述べており、合成された楽音やデジタル入力 (サンプリング) された自然音や音声などに対する信号処理 (プロセッシング) に関しての考察は、次節に譲る。

また、楽音合成方式の中で「物理モデル」方式と呼ばれる、ウェーブガイド (Waveguide)、Karplus-Strong などの楽音合成方式についても、「楽音と音場のシミュレーション技術」という視点から次節で統一的に扱うために、ここで紹介する分類の中からは除外している。

1.2.1 楽音合成の要素技術

本節では、いろいろな楽音合成方式を技術的に理解する上での助けとなる、コンピュータ音楽における要素技術のいくつかを紹介する。

一般的基礎知識

まず、本報告書では触れるまでもなく省略した一般的な電子技術の知識として、コンピュータ音楽における楽音合成・信号処理に関連する基礎的な技術項目を列記しておくと、以下のようなものがある。

1. アナログ — デジタル変換
2. デジタル — アナログ変換

3. サンプリング定理
4. 離散化と量子化の原理

これらの基礎知識については、コンピュータ音楽に限らず、一般的なデジタル信号処理技術に関する文献のほとんど全てにおいて述べられている。

ユニットジェネレータ (Unit Generator)

ユニットジェネレータとは、楽音合成のアルゴリズムの基本となる単位発振器のことで、図1.1左のように2つの入力 [inc][amp] と1つの出力 [out] からなり、内部に特性テーブル [Table] を持っている。

ユニットジェネレータのこれらのパラメータは、一般に「音の3要素」と言われる

1. ピッチ (Pitch: 音の高さ)
2. ラウドネス (Loudness: 音量)
3. ティンバー (Timbre: 音色)

に対応している。すなわち、

inc 波形テーブルを繰り返す読み出すためのアドレスの進むスピードに相当する。結果として楽音信号のピッチに対応する。

amp 発生される楽音信号の振幅に相当する。結果として楽音信号のラウドネスに対応する。

Table 内部にデジタル的に用意される波形テーブルで、システムに固有の量子化ビット数とアドレス空間をもつ。このデータによって周期的信号波形が与えられ、結果として音色に対応する。

という関係をもつ。このようにユニットジェネレータが楽音信号などの周期的信号の発生に用いられる場合には、「テーブル参照発振器」(Table Look-Up Oscillator) とも呼ばれる。

変調要素としてのユニットジェネレータ

ユニットジェネレータは楽音信号そのものを発生する部分だけでなく、楽音合成のアルゴリズムに必要な各種の信号発生器としても活用される。これは変調 (Modulation) 要素としてのユニットジェネレータの利用である。

たとえば図1.1右のように、ピッチを発生するためのユニットジェネレータの [inc] パラメータ入力に対して、別のユニットジェネレータの出力を与えることは、後述の周波数変調 (FM、あるいはビブラート) に相当することになる。

時間要素としてのユニットジェネレータ

ユニットジェネレータは内部テーブルのデータを読み出すデータ発生器なので、通常はテーブルデータの全域を1周期として繰り返す、周期的な信号の発生器である。しかしこれを周期的でなく、たとえばテーブルの最後まで行ったところでストップするようにして、「ノート・オンの際に1回だけデータを読む」ように設定すると、そのまま楽音の時間的変化特性データ (エンベロープ Envelope) の発生器となる。

たとえば、ユニットジェネレータをこのようなワンショット (One-Shot) 動作に設定して、ピッチ発生のユニットジェネレータの振幅パラメータである [amp] 入力に供給すると、一般にエンベロープと言われる楽音の音量の時間的変化特性を実現できる。

なお、エンベロープは音量に限らず、たとえば

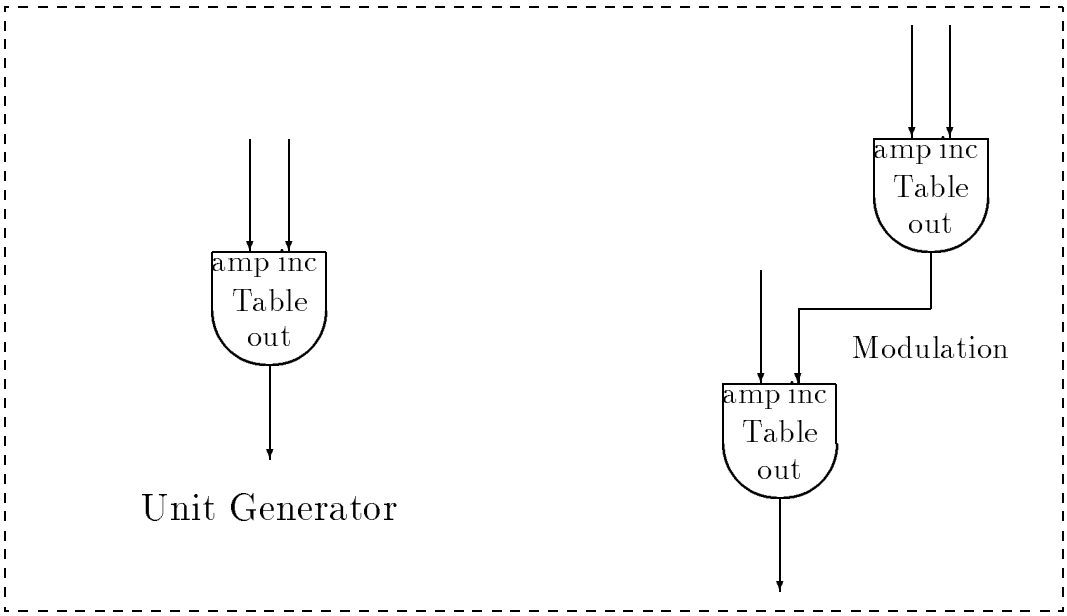


図 1.1: ユニットジェネレータ

- 倍音成分の強度変化→音色変化
- フィルタのカットオフの時間変化→音色変化
- 変調の強度変化→音色変化
- ピッチの緩やかな時間変化→ビブラート
- 振幅の緩やかな時間変化→トレモロ
- 音程の時間変化→音列生成 (アルゴリズム作曲)
- パンポット情報の時間変化→音像移動

など、コンピュータ音楽のいろいろな要素として活用されている。

ユニットジェネレータの実現方法

ユニットジェネレータとは、あくまで楽音合成アルゴリズムを検討するための理想的 (ideal) な概念でしかない。Computer Music の研究者がコンピュータ上に「楽器」を実現する場合には、ユニットジェネレータの動作をプログラム (モジュール) として記述して、発生する楽音信号に相当する出力を計算によって求めて、最終的に D/A コンバータで音響に変換すればよい。いわば「ソフトウェア楽器」である。

しかし、電子楽器のような具体的なハードウェアとして音楽演奏に使う場合には、リアルタイムに楽音出力を得ることが必要であり、ごく最近までは「ソフトウェア楽器」の概念であるユニットジェネレータは現実的なものではなかった[2]。

そこで、実際のハードウェアシステムとして楽音合成を実装するためには、基本的には以下のような方法をとってきた。ここではこれ以上詳しく触れないが、これはいわば「電子楽器の歴史」に相当する。

アナログ方式　ユニットジェネレータの [inc] をアナログ発振器の周波数に、[amp] を可変増幅器のゲインに対応させる。

ディジタル方式　ユニットジェネレータの [inc] をメモリ読み出し速度に、[amp] をエンベロープ乗算値として対応させる。

[2]いわゆる DSP 技術がこれを可能にした。IRCAM の ISPW が有名である.

DSP 方式　ディジタル方式のなかで、マイクロプログラム方式の信号処理プロセッサ (DSP) を活用して、半ソフト的に実現するもの。

ソフトウェア方式　ユニットジェネレータをソフトウェア的なモデルとして記述し、最終的な処理結果まで演算して D/A 出力するもの。

演算子要素 (Modifiers)

ユニットジェネレータの考え方で楽音合成のアルゴリズムを構成していく場合に、数学的・電子回路的な構成要素として、さらに以下のようなものも使われる。

加算器　楽音信号の線形性を利用して、いろいろな信号を重ね合わせる場合に使われる。ディジタルの時分割演算のための累算器としても使用される。

乗算器　楽音信号の線形性を利用して、2 種類の信号の変調・比例関係などのために使われる。具体的に構成する場合には、固定小数点・浮動小数点のそれぞれの方式が検討される。

変換テーブル　おもに非線形関係を記述する場合に、ディジタルシステムであれば変換テーブルを任意の精度で用意して参照する。ハードウェアで実現していく場合にも、ROM や RAM をシステムに応じて使い分けている。

加算器と乗算器というのは、DSP(Digital Signal Processing) 技術の中心要素となるものである。ハードウェアとして実現される場合には、離散化した時間 (タイムスロット) ごとに所定の精度の演算を終了するために、高速な専用回路として構成される。ソフトウェア楽器の場合でも、演算誤差が累積することを避けるために、十分なデータ長として表現される。たとえば、最終的な出力 D/A コンバータのビット精度は一般に CD や DAT と同じ 16-20 ビット程度である場合が多いが、ディジタル楽音合成システムの内部演算部分では、32 ビットから 40 ビット程度、場合によっては 48 ビット精度以上の処理を行っている。これはワークステーション級の上級システムばかりでなく、低廉な民生用電子楽器に使われる音源 LSI 内部においても同様である。

1.2.2　加算型

ここでは、ディジタル楽音合成方式の第 1 のタイプとして、「加算型」と呼ばれる一群の方式として、

1. サイン合成方式
2. 分析・再合成方式
3. FFT 方式
4. ボコーダ方式

をとりあげる。なお、これらは原理としてはかなり歴史のあるものなので、それぞれ関連する最近の研究の報告からトピックを拾ってみた。

サイン合成

人間の聴覚は「音色」を倍音 (高調波スペクトル) の組み合せとして知覚することから、楽音合成の原理に自然な方式として、サイン合成はもっともポピュラなものである [Moo90]。

ユニットジェネレータでサイン合成を実現するためには、発生する楽音の基音から 2 倍音、3 倍音... という整数倍音のそれぞれに相当するピッチ [inc] のユニットジェネレータを用意して、それぞれの [amp] には別個のエンベロープ発生のためのユニットジェネレータからの出力を供給する。これらの倍音成分をすべて加算することで、最終的な楽音信号が得られる。

ハードウェアとして実現するためには、1音を発生するために倍音数だけのジェネレータを必要とし、時分割多重化させたとしてもシステムの規模はかなり大きなものとなる。しかし実際に、すでに64倍音とか128倍音までの全ての高調波に独立のエンベロープを持たせた楽音合成システムは、専用LSIによって民生の電子楽器として実現されているために、このレベルはもはや研究の対象ではなくなっている。

最近の研究としては、定常的な音色でなく非常に短時間の離散フーリエ変換DSTFT(Discrete Short-Time Fourier Transform)によって、音声のような非定常的な音響や、自然楽器においてある音から別の音に移る瞬間の複雑な変化を発生する部分が注目されている[Str89]。

また、音声分析の手法であるMQ(McAulay-Quatieru)法を適用して、このような複雑に変化する楽音を分析・合成する方法も研究されている[HH92]。この場合、楽音合成に力点を置けば一種のサイン合成方式の展開形ともいえるが、次のAnalysis/Resynthesis方式の応用とみることもできる。

分析・再合成 (Analysis/Resynthesis)

定常的な楽音信号にFFTをかけてスペクトル成分を抽出し、このデータに基づいて高調波係数としてサイン合成の重みづけをすれば原理的には再合成される。つまり前述のサイン合成方式とは、ここで述べる分析・再合成方式のもっとも典型的な一手法でもある。

一般に分析・再合成方式では、

1. 分析によるパラメータ抽出
2. パラメータ処理 (クロス・シンセシス)
3. パラメータによる再合成

というステップによる。これは見方を変えると、パラメータという形態で一種の情報圧縮が行われていることになり、データ転送量の限定された通信回線で多量の情報を忠実に転送するための「情報通信技術」の分野と関係したものである。

分析・再合成方式を、特徴となるパラメータの抽出方法として列記すれば、

1. サイン合成 (倍音成分) による方法
2. 線型予測LPC(Linear Predictive Coding)による方法
3. 調和成分とノイズ成分を分離する方法
4. フォルマント関数を抽出する方法
5. バンドパスフィルタ群としてモデリングする方法

などがあり、関連する文献も[Rod89]に紹介されている。

この方式に関する最近の研究の一例としては、オーバーラップさせたサイン関数表現による分析・再合成システムによって、ピッチによらず自動的に分析・再合成する手法が紹介されている[BS91]。

FFT

FFTはすでに一般的なディジタル信号処理の技術として定着し、計測器やパソコンでの音声分析などに広く活用されている。また、楽音合成におけるFFTは、前述の分析・再合成方式などでも広く利用されている。(名前としてよく使われるので独立させたが、実質的にはサイン合成方式と変わりがない。)

最近の研究としては、音楽のリアルタイム演奏、あるいは演奏者とのインタラクティブなやりとりを楽音合成の領域から実現するために、リアルタイム処理による短時間FFT(SFFT)を活

用して各種パラメータを抽出するとともに、「確定的」な成分と「確率的」な成分とで別個に合成し、「音色のレベルからリアルタイムコンポジションしていく」アプローチがある [SS89]。

ボコーダ (Phase Vocoder)

バンドパスフィルタ群によってフォルマントフィルタのパラメータを抽出し、これを再びバンドパスフィルタ群のゲイン特性として再現したアナログボコーダ以来、ボコーダの独特の音色は音楽に広く使われている。一方、通信技術においては音声情報圧縮の手法としてボコーダ技術がはやくから研究されており、コンピュータ音楽の技術がディジタル化されるとともに、この両者が結びついて、ボコーダ関係の楽音合成、あるいはボコーダ的な楽音信号処理がいろいろと検討されてきた。

基本周波数を整数倍音系列にとらない、という条件をのぞけば、基本的にボコーダは一種の FFT でしかない [Moo90]。このためボコーダ自体の研究というよりも、実際の音楽制作の現場での適用事例として、マルチプロセッサのネットワーク環境でボコーダ処理を分散させる手法など、応用的な研究に関連して登場している [KKOH92]。

1.2.3 減算型

ここでは、ディジタル楽音合成方式の第 2 のタイプとして、「減算型」と呼ばれる一群の方式として、

1. ディジタルフィルタ方式
2. 線型予測方式
3. フォルマント方式

をとりあげる。なお、これらの分類はかなり意見の分かれるところで、人によっては「加算型」に分類されているものもある。

減算型の原理としては、アナログシンセサイザの VCF がその源流にあり、実際の電子楽器としてはサイン合成方式よりも古い。また、物理モデル方式を待つまでもなく、人間の「声」にしても、声帯からの音源信号を咽喉・咽頭のフィルタで調整する、という一種の減算方式であるために、人間の感覚に親しい楽音合成方式でもある。そこで、このタイプの楽音合成方式のキーワードとして「フィルタ」に注目することが重要である。

ディジタルフィルタ

アナログシンセサイザの VCF は一般に LPF(Low Pass Filter) であり、これを単純にディジタルに置き換えて実現したのが、ディジタルフィルタ方式である。具体的には、サンプリングに対応した演算周期ごとに

- システムクロック単位の遅延要素
- 係数のための乗算要素
- 累算のための加算要素

を用意して多段結合すれば、IIR タイプないし FIR タイプのディジタルフィルタが構成される。

リアルタイムに稼働する現実のシステムとして実装する場合には、アナログのように発振すれすれまでレゾナンスを上げることが難しいとか、量子化誤差や演算誤差によるノイズなどが発生するために、アナログのフィルタ並に「自然な」フィルタにするのは、原理的にもかなり困難

である。さらに、遅延要素のために原理的に発生するレスポンスの遅さ、エンベロープによって時間的変化させる場合のパラメータ選定の困難さなど、多くの課題をもっている。

また一方では、ディジタルフィルタを理論通りに構築できた場合の「不自然さ」を検討し、敢えて特性に歪をもたせて「暖かいディジタルフィルタ」を実現する、という報告もある [Ros92]。このタイトルの「ディジタルフィルタをアナログのように聴かせる」という考え方自体、ディジタルはアナログよりも優秀である、という一般的な概念が音楽の世界では否定されうる事実を示しており、結局は「人間の聴覚 (心理) に回帰するコンピュータ音楽」の重要な視点として興味深いものがある。

線型予測

線型予測 LPC(Linear Predictive Coding) による情報圧縮 (特徴抽出) がコンピュータ音楽の世界で楽音や音声の分析・合成に適用されるのは、情報理論的に「楽音・音声という情報は冗長である」という性質があり、これを利用しているからである [Moo90]。

LPC 方式の楽音合成もボコーダ方式やフォルマント方式と同様に、定常的な部分の多い楽器音よりも、より劇的に変化する「声」の合成に活用される場合が多いが、アプローチとしてはより音声処理 Speech Processing の技術に近く、ベル研究所など通信関係の研究者が中心となってきた。

LPC によって抽出された各種の楽音パラメータにもとづいて楽音合成を行う方法としては、

1. スペクトルモデルに適用する
2. 全ポールフィルタに適用する
3. ウェーブガイドのループフィルタに適用する

などの方法がある。つまり見方を変えると、LPC 合成方式というのは音響的に良好な、短時間パワースペクトルを非線型にスムージング (なめらかに接続) する方式、ということができる [Smi91b]。

フォルマント (Vowel)

後述する PCM 方式の最大の欠点は、サンプリングされた楽音はその周波数ではリアルであっても、別の再生周波数に対しては本質的に不自然になってしまうところにある。これは自然楽器や音声の発音原理 (音響物理) からすれば当然で、固有のレゾネータである共鳴体による効果は、楽音のピッチに関わらず固定していなければならないのは当然である。

フォルマント方式はまさに、この楽音固有の特性をフォルマントフィルタとして実現したものであり、アルゴリズムによる楽音合成方式と「物理モデル方式」との橋渡しをする役割りにある。

コンピュータ音楽におけるフォルマント方式の適用は、おもに「声」を合成させる方向を指向しており、線型予測 LPC、ボコーダなどとともに、フォルマント・トラッキングによって「歌うコンピュータ」を実現するための研究が続けられている [Dod89]。

「声」は固有のピッチをもつ複数のフォルマントフィルタを用意すれば済むものではなく、たとえば母音ごとに各フィルタの中心周波数を変えなければならず、さらに女性の声と男性の声でもパラメータが変わるなど、図 1.2のようにアルゴリズムはかなり複雑になる [Moo90]。

また、IRCAM で研究・開発されている CHANT というボイス・シンセサイズ方式 [Ben89] や、FOF というボイス・シンセサイズ方式 [CMBP92] も、フォルマント方式の一種のバリエーションということができる。

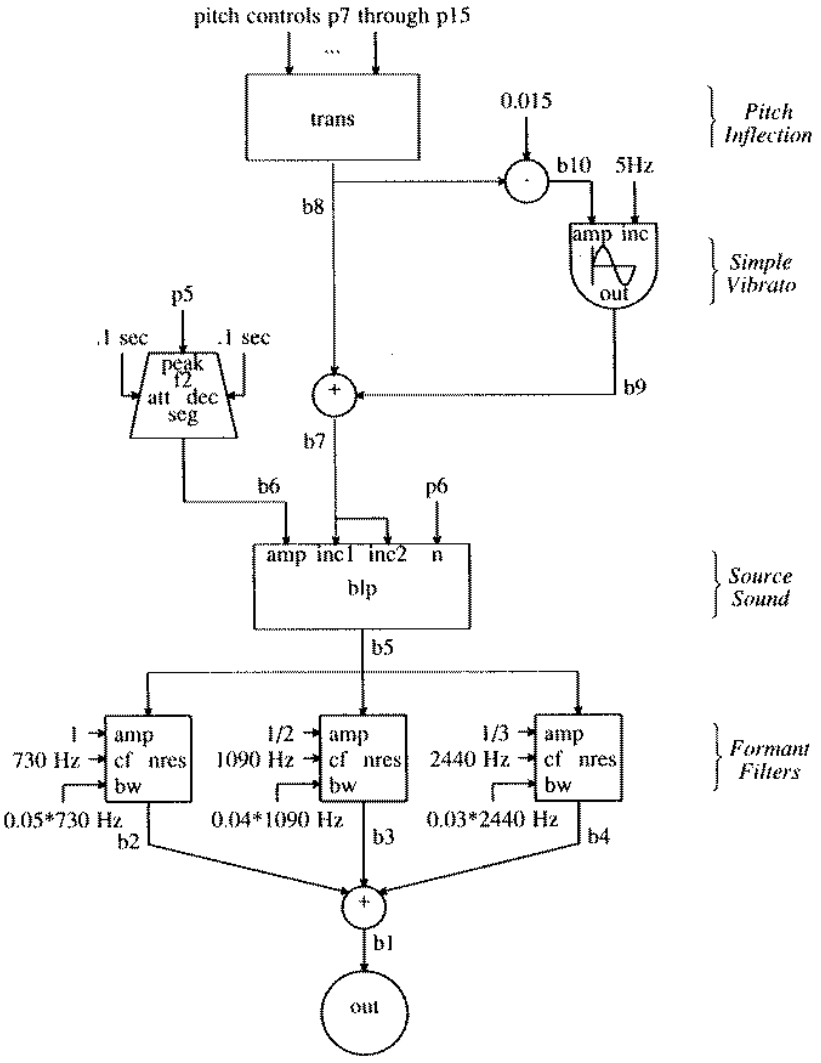


図 1.2: Cmusic Instrument for Producing the Sound of a Male Voice Singing /a/

1.2.4 非線形変換

ここでは、ディジタル楽音合成方式の第3のタイプとして、「非線型変換型」という視点からまとめて

1. 振幅変調方式
2. 周波数変調方式
3. ウェーブシェーピング方式

をとりあげる。つまりこれらの方式は、楽音を発生するユニットジェネレータの持っているパラメータのうち、順に

1. ユニットジェネレータの [amp] を変調するもの
2. ユニットジェネレータの [inc] を変調するもの
3. ユニットジェネレータの [Table] を変換するもの

と考えることができる。

振幅変調 (AM)

ピッチとともに音量は楽音のもっとも基本的な要素であり、初期のアナログシンセサイザの時代から、音量エンベロープとともに、定常的な振幅変調として、LFO(Low Frequency Oscillator) による「トレモロ変調」が行われてきた。

トレモロと知覚されるのは数ヘルツまでのオーダであり、LFO の周波数を 100Hz オーダに上げると、むしろ音色を変化させる変調として知覚されるようになる。さらに、楽音発生のピッチに対応させて、ピッチと同じ周波数レンジのジェネレータによって多段に AM をかけると、ユニークな音色を発生することができる [Moo90]。

また、通常は AM はゼロから 1 までの変調度をもっているが、これをバイポーラ (両極) としてスイッチングしたものが「リング変調」(平衡変調) である。これは通信分野ではよく使われる技術で、キャリヤ (搬送波) 成分が打ち消し合ってしまうために、元の楽音信号のピッチ成分や倍音成分とはまったく異なる高調波成分となり、この独特の音色も音楽に活用されている。

周波数変調 (FM)

民生のシンセサイザと簡易型パソコン内蔵音源として定着した FM 方式 [Cho73] については、コンピュータ音楽の研究者の興味の対象としてはすでに過去のものとなった感がある。

もともとサイン合成では作りにくい非整数次倍音を容易に発生できることと、サイン合成やディジタルフィルタよりも実現する回路規模がコンパクトであるところが FM 方式の最大のメリットだった。ところが半導体の集積度向上による大容量メモリ LSI の時代に入ると、自然楽器音の模倣では PCM 方式のリアリティにかなわず、また安価な FM シンセサイザ音があまりに普及してしまって、コンピュータ音楽の音素材としての魅力は激減してしまった。

最近の研究としては、もっぱら FM は他の楽音合成アルゴリズムと比較するために引合いに出されることが多く [HBH92]、その多くは単純な FM 方式のデータを「過去の世代の楽音合成」として比較対象に利用している。

ウェーブシェーピング (Waveshaping)

ウェーブシェーピング方式は、基本的には「波形を歪ませる」ことであり、ディジタル的に具体化する場合には「変換テーブルを参照する」ことに尽きる。

ただし、楽音合成に利用する目的はいろいろあり、

- FM のようにとにかく「ヘンな音」を作りたいから
- パラメータの補間手段として活用する
- システムとしてインプリメントするのが容易だから

などの動機によって、実際のシステム形態はかなり変わってくる。

たとえば [BH92] の場合には、自然楽器の特徴をシミュレートする手段として、時間・音量・音域などのパラメータに対応してなめらかに歪が増加・減少するようなテーブルを楽音の各パーシャルごとに用意することで、少ないパーシャルでリアリティの良好な楽音を発生している。これは電子楽器のように一定の制限のあるシステムでは有効な手法であるといえる。

1.2.5 サンプリング

ここでは、ディジタル楽音合成方式の第 4 のタイプとして、いわゆる「PCM 方式」に代表される、サンプリング方式について考察する。

もともと楽音合成でユニットジェネレータのような「周期波形発生器」が使われたというのは、楽音信号を最初から最後まで繰り返さずに表現するには、膨大なデータ量が必要とされる、

という制約があったからである。これを情報圧縮するために、「繰り返し」部分の冗長性に着目して周期的信号を発生し、ここに非周期的 (時間的) 変化をいかに付与していくか、という各種の楽音合成技術が研究された。

ところがコンピュータ技術・LSI 技術の進展は、音響信号を冗長であろうがなかろうが「そのままディジタル記憶・再生できる」状況になってきた。たとえば CD には、44.1KHz サンプリング 16 ビット量子化で数十分間の長さの音響信号が格納できる。コンピュータの内部メモリを多量に用意してこれを展開すれば、周期的に扱わなければならない、という制約を受けずに音響信号が任意に処理できることになった。

いろいろな回路技術を駆使してきた民生の電子楽器の世界ではさらに顕著にこの影響が現れてきた。もともと一般の電子楽器は、自然楽器の音を模倣することが重要な目標であり、このためには大容量低価格のデータ ROM を多数搭載して、ディジタルレコーディングされた「本物と変わらない」音をそのまま「再生」すれば、目標は容易に達成されてしまう。かくして、メモリ物量作戦によるサンプリング方式の全盛時代になった。

そして、コスト条件の厳しい電子楽器の世界では、「パーシャル」と呼ばれる単位ジェネレータを複数個用いて、「楽音の立ち上がり部分の非定常的な音をワンショットだけ再生」する部分と、「楽音の定常的な部分を周期的に再生」する部分とで合成する、という方法をとるのが一般的になっている。さらに音色の時間的変化やスペクトル補間として、クロスフェードによるミキシング手法 [SRD88] を用いることも多い。

ところで、コンピュータ音楽の研究者や作曲家にとって、楽音合成としてのサンプリング方式はほとんど興味の対象ではない。音楽の要素として「音色」を扱う立場としては、「模倣しかできない」方式は対象外なのである。そこでこの方式は「音響を録音・再生する道具」として、楽器や音響機器として使うのみで、楽音合成としての処理は Signal Processing のステップで行っていくことになる。

1.2.6 グラニュラーシンセシス

ここでは、ディジタル楽音合成方式の第 5 のタイプとして、グラニュラーシンセシス Granular Synthesis を紹介する。これはアイデアとしてはかなり古いユニークな楽音合成方式で [Roa78]、いわば「音の量子力学」である。

すなわち、「音」と知覚できない (周期振動のない) 微小でなめらかな音圧変化パルス粒子 (Grain) を非常に多数個用いて、ある種のランダムネスをもって時間的空間的に配置する、というものである。それぞれの Grain の形状や幅、時間的平均密度、空間的定位、音量分布などによって、発生して知覚される音響はさまざまに変化し、これらが楽音合成のパラメータとなる。

グラニュラーシンセシスの原理はとても簡単なものだが、たとえば各 Grain を発生して 1 秒間に数万個ランダムに配置する、という計算処理はかなりの規模のため、大型計算機による非リアルタイム計算しかできなかった第 1 次世代は一定の段階で留まっていた。これが最近の EWS のパワーアップ等によって、リバイバルの第 2 次世代を迎えた。

ここではワークステーションのグラフィカルなヒューマンインターフェースを活用したパラメータ制御 [OHK91] や、ニューラルネットワークを利用したリアルタイムのパラメータ補間 [Nag92] など、原理としてはすでに完結しているグラニュラーシンセシスを、全体として音楽にどう利用していくか、という研究に移っている。

グラニュラーシンセシスには「ピッチ」の概念がないために、「時間的に音程を配置する」という作曲でなく、「時間的に変化する音響をパラメータの変化として全体として記述する」ことが作曲となる。このために、リアルタイム処理能力の向上したコンピュータを利用することで、第 2 次世代のグラニュラーシンセシスは、ようやく Interactive Composing の段階に入ったとも

いえる。

1.2.7 楽音合成技術の展望

コンピュータ音楽が「音」を基本的な素材としていく限り、楽音合成は今後もコンピュータ音楽の重要な一部門であり続ける。ここでは、楽音合成技術のこれからの展望について考察する。

今後のコンピュータ音楽の課題として、次の3種類のギャップをどう橋渡ししていくべきか、という問題提起があった [Ris89]。

1. Computer Sound と Instrumental Sound
2. Real-Time Synthesis と Delayed Synthesis
3. Digital Synthesis と Digital Processing of Natural Sound

このうちまず第1点は音楽美学に関する問題なので、ここでは触れない。また第2点については、これまで非リアルタイム処理でしかできなかった楽音合成が、間違いなくリアルタイム化されていくことになる。コンピュータパワーの向上とともにヒューマンインターフェースやVR技術によるセンシングシステムが進歩すれば、インタラクティブ性が確実に向上して、音楽の「演奏」の考え方が変わる。つまり、「すでにある音を鳴らす」のでなく、「鳴っている音を変える」という演奏が可能になり、自然楽器にまた一歩近づくことになる。そして第3点は、現在着実にギャップが埋まりつつあり、この両者は区別なく音楽の素材として活用される時代がすぐ間近になってきている。

コンピュータ音楽というのは、電子回路技術とコンピュータ技術に大きく影響されるもので、この性質は現実のシステムではなおさら顕著である。たとえばディジタル方式の楽音合成は、電子楽器という装置として提供されるためには、コストを考慮して専用化されたLSI(いわゆる「音源チップ」)として、はじめて実現された。ここには音楽家どころか、研究者も手を出せない「LSI経済(ビジネス)の壁」があった。コンピュータでソフトウェアが計算する「楽音合成アルゴリズム」では、現実に「楽器」としてリアルタイム発生するには処理速度が1000倍ほど遅いが、音源LSIを工業生産するために自由な楽音合成のための機能を大幅に削減して専用化した、楽器メーカ仕様のチップしか選択肢がなかったのである。

ところがディジタル信号処理の技術が一般化して、汎用のDSPチップが提供されるようになると、状況は研究者のために好転した。コンピュータに演算エンジンとしてDSPボードを組み込み、DSPのマイクロプログラムとして楽音合成アルゴリズムを記述すれば、原理的には誰でもオリジナルな楽音合成システムを構築できる可能性が生まれたのである。ただし、汎用DSPは一般に単純な演算器を持つだけの「裸のハードウェア」であって、固有のマイクロプログラミングは音楽家に手の出せるものからほど遠い、相当にエンジニア寄りのものでしかなかった。

そして時代はさらにソフトウェア化を進めている。コンピュータに組み込む信号処理エンジンボードは、IRCAMのISPWにおいてDSPでなくRISCチップを採用することになった。これはリアルタイムの世界でも「楽音合成のソフトウェア化」が始まったことを意味する。DSPハードウェアに固有の表現でなく、RISC用コンパイラがエンジニアリング寄りの処理をブラックボックスの内部に隠して、さらにこの上の階層にMAXライクなGUIが置かれて、ようやく音楽家が抵抗なく楽音合成や信号処理のアルゴリズムを直接にプログラミング(作曲)できる環境が登場した。

ここにきて、楽器メーカやDSPチップメーカの論理と離れて、自由に楽音合成をプライベートに研究・実践できる時代が到来した。コンピュータ技術の進展はこの環境をより良好に、低廉に、多くの芸術家に提供していくだろう。マルチメディア技術によって映像情報と音響情報がリンクして、やがては一般のエンターティメントとして、CGと同期してリアルタイムに進行する

楽音合成が実現していくものと思われる。そしてその時代になってもコンピュータ音楽の世界で変わらず重要なのは、

- ユニークなアルゴリズムを創造する発想の自由さ
- 「合成音」楽器の永遠の理想である自然楽器との共存

であろう。

1.3 楽音と音場のシミュレーション技術 (藤森)

CMJ[3]やICMC[4]のプロシーディングに発表された論文を中心に、コンピューター音楽に利用される技術とその動向を特に実時間シミュレーションという観点から展望することを試みる。前の章において、分析と合成という観点から楽音について述べられているが、それとは異なったシミュレーションという観点を採用することで、楽音を直観的あるいは経験的に構成する手段を与える技術について述べる事ができるだろう。シミュレーションは最近話題に多く取り上げられている、AR[5]とかVR[6]と呼ばれる概念に関係する技術と言えるが、このような概念は個別の技術を統合する手法の開発を促す側面を持つだろう。特に音場の技術はVRの概念との関係から述べられる機会が多いようであるが、音楽にもそういった概念を導入しようという試みもある。これから、楽音と音場のシミュレーションによって支えられる「世界」モデルの例をあげ、その後その構成要素それぞれのシミュレーションについて概観する。これらコンピューター音楽に応用される技術の学術的、あるいはより専門的な背景についても触れるつもりであるが、筆者の能力不足による不適切な記述などもあろうかと思うがお許しいただきたい。

1.3.1 ワールド・モデル − 楽器、演奏者、音場、聴取者

基本的なシミュレーションの対象世界は、楽器、演奏者、音場(特に室内音場)、聴取者 と設定しよう。そして、これらの物理的(音に関係する)側面の実時間シミュレーションについて考える。以後特に断わらないかぎり、「シミュレーション」をこのような意味で使用する。この様な日常的とでも言えるものをベースにする理由は、技術的詳細に立ち入らないでも利用者が理解しやすいと思われるからだ。これから先において、各対象あるいは対象間の相互作用に関するシミュレーション技術について考察するわけだが、コンピューター音楽システム全体としては日常的、あるいは音楽的言葉で理解しうるように構築されるのが望ましいと考える。

さて、ここに挙げた四つの構成要素とその相互作用について考える場合に問題となるのは、演奏者と聴取者である。この二つの対象は人間であるため、他の二つに比べて取り扱いが複雑で、どのような側面をシミュレーションするのかを考慮する余地が多い。ここで考えているシミュレーションは、音の合成に関するものに限定しているので、演奏者については楽器と相互作用する運動系の一部、聴取者については耳の近傍の音響特性をそれぞれの代表とする。このように、音の合成方法をより具体的にはデジタル信号処理技術(あるいは離散数値シミュレーション)からそれらを見ていくが、演奏者、聴取者に関する最も興味深い技術はデジタル信号処理技術から離れ、それぞれロボット制御工学や音響心理等、別の分野に移って行くだろう。特に楽器との相互作用が密である演奏者に関する記述は、デジタル信号処理からは逸脱するが、ある程度は止むを得ないと考える。ただ、この辺の問題のより専門的な取り扱いは、インターフェースに関する別の章でなされると思われる。

---

[3]Computer Music Journal

[4]International Computer Music Conference

[5]Artificial Reality、人工現実

[6]Virtual Reality、仮想現実

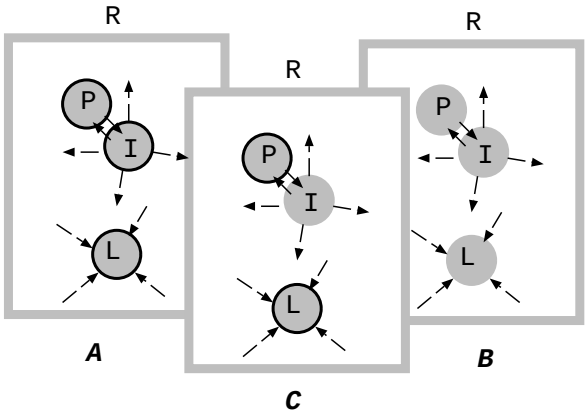


図 1.3: ワールド・モデル

ここに文字 I, P, R, L は楽器、演奏者、音場、聴取者を示す。A は現実、B はシミュレーション、C は A、B の各要素を組み合わせた場合を表している

ワールド・モデルの四つの構成要素は、力学的あるいは音響的な関係で結ばれている。

図 1.3は、矢印で書かれている変換の部分を適当に設定すれば、現実A とシミュレーション B の構成要素を混ぜ合わせる事が可能だという事を示している。いうなれば、置き替えによるクロスシンセシスに相当する事ができると言えよう。既にこのようなプラガブルな状況はすでに一般的になっており、今後ますます多様化するであろう。実際に電子楽器を使用した演奏は、”楽器”と場合によっては”室内音響”に相当する部分がシミュレーションされ、キーボードやスピーカー等によって現実の演奏者や聴取者に結合されている。

又、室内音響に対しては最近ではアクティブ制御技術の開発がさかんであり、現実の音響空間特性そのものを可変することも可能になってきている。

シミュレーションのためのモデル化は、各要素によって幾分異なる。その差異は本質的なものではなく、あくまでも必要とされる精度や実装方法の技術的制約によるものである。たとえば、楽器については、内部の音の伝播の状況までシミュレーションできるようになってきているが、音場の場合はほとんどが、その系のインパルス応答のシミュレーションによっている。前者の場合も、演奏者との相互作用 (奏法の違いなど) を無視できたり、形状の変化を考慮しないならば、そのような詳細なシミュレーションは不要となる。あるいは、実装段階で冗長度を減らすなどの工夫もできる。後者の場合は、詳細なシミュレーションを実時間に行うのが困難であるし、その技術的背景にある建築音響関連において、音響空間と人間とのダイナミックな相互作用があまり重要視されていないため、あまり多くの技術が開発されていないのが原因かも知れない。

さて非常に簡単に言えば、物理シミュレーションの特質は次のように考えられる。

1. 音楽家にとって従来からの経験が利用できるので、扱い易い。
2. 楽器の物理シミュレーションに取り組むと、自然に演奏者を部分的ではあっても同時に取り扱う必要がある。
3. 個々のモデルの精密さを追求すると、汎化が困難になる。

1.3.2 楽器 – 演奏者系

自然楽器の基本的性質の研究は音響学の領域で主に行われており、音響学的な研究の流れはそれに関係する学術書、学会誌を参照されたい。学会での研究発表に関しては、[中村 89b, 久保 91, 中村 90b, 安藤 89, 中村 89a, 藤崎 89, 岩宮 86, 久保 86, 中村 86] などで紹介されている。

表 1.1: デジタル合成技術の分類

Processed Recording	**Spectral Model**	**Physical Model**	**Abstract Algorithm**
Concrète	Wavetable F	Ruiz Strings	VCO, VCA, VCF
Wavetable T	Additive	Karplus-Strong Ext.	Some Music V
Sampling	Phase Vocoder	Waveguuide	Original FM
Vector	PARSHL	Modal	Feedback FM
Granular	Sines+Noise (Serra)	Cordis-Anima	Waveshaping
Prin. Comp. T	Prin. Comp. F	Mosaic	Phase Distortion
Wavelet T	Chant		Karplus-Strong
	VOSIM		
	Risset FM Brass		
	Chowning FM Voice		
	Subtractive		
	LPC		
	Inverse FFT		
	Xenakis Line Clusters		

一般的に言って音響学での自然楽器の詳細なモデルは、その楽器の音を特にリアルタイムに合成するには計算量が多く実用的なシステムを構築するのは容易ではない。さらに、物理モデルによるシミュレーションには、演奏者とのインターアクションが重要であるために、リアルタイムでの音の合成が望まれる。この様な観点からモデルの簡略化や効率良い実装方法が検討される。また、音楽では複数の楽器が使用されるために、シミュレーターも複数あるほうが都合がいいので、この点からも効率が重視される。結局のところモデルの精密さと実装効率のトレードオフの問題であるが、音楽の中で使用される場合には、現実に忠実なモデルでない方が音楽的に魅力のあるものと考えられる事もありうる。

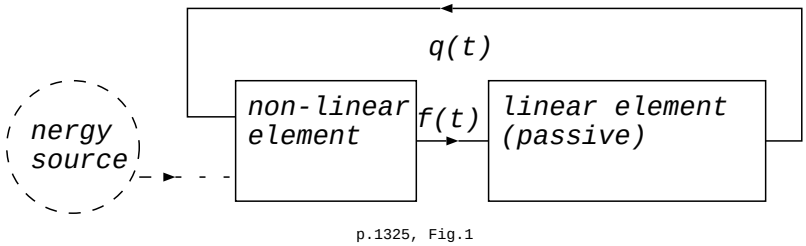
楽器

色々な分類が提案されている中で最近のデジタル楽音合成について、 Smith は表 1.1 [Smi91b, p.6] のように分類している。

物理モデルの構築の出発点では、聴覚による観察よりはむしろそれ以外の視覚等による観察から得られる情報に重点がおかれていると考えられる。 Smith[Smi91b, p.9] も述べているように、楽音合成技術の流れは発音点、すなはち楽器の観察による物理モデルと受音点、すなはち聴覚の観察によるスペトルモデルの二つに集約されると言えよう。

物理モデルによる楽音合成の先駆的研究は、 1971 年頃 Ruiz らによって行われた [HR71a, HR71b]。彼らは振動する物体を微分方程式で表現し、それをデジタルコンピューターで繰返し計算によって解く手法を用いた。しかしながら、微分方程式といった数学的な表現で複数の物体の相互作用による発音 (例えば、弦と弓) を扱うのは困難である。

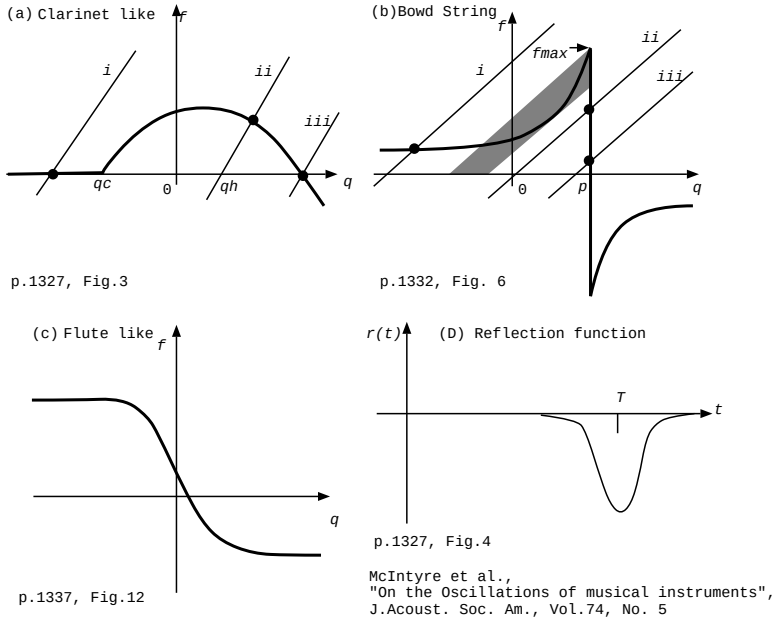
近年の物理モデルをベースにした楽音合成の研究は、私が知る限りでは、 Cadoz あるいは Adrien ら仏国勢によるものと、 Smith, Cook ら米国勢による二つの大きな流れがあると言えよう。 Cadoz らの研究は、あとで演奏者の項で取り上げるが、演奏者と楽器とのインターアクション、あるいは演奏者のジェスチャに重点を置いたかなり広い展望に基づく研究であって、自然楽器のシミュレーションにはあまり深くかかわっていなかった。複数の質点 – バネ系を構成要



p.1325, Fig.1

McIntyre et al.,
"On the Oscillations of musical instruments",
J.Acoust. Soc. Am., Vol.74, No. 5, 1983

図 1.4: 楽器振動のブロックダイアグラム



p.1327, Fig.3

p.1332, Fig. 6

p.1337, Fig.12

p.1327, Fig.4

McIntyre et al.,
"On the Oscillations of musical instruments",
J.Acoust. Soc. Am., Vol.74, No. 5

図 1.5: 非線型特性、反射関数

素とする直接的なモデルを採用しており、最近の Adrien らによるモーダル合成方法に発展していく。一方 Smith らは、おそらくいわゆる Karplus–Strong アルゴリズムの発見がきっかけとなり、非線型回路を含む遅延回路ループを基礎とする効率的な自然楽器のリアルタイムのシミュレーション (楽音合成) の開発へ向かった。

McIntyre, Schumacher と Woodhouse らによる論文 [MSW83] は、効率的なシミュレーション方法で、しかも多くの種類の楽器に適用可能なモデルについてまとめられている。そのモデルを基に、まずクラリネットを例に取りあげ、その発音機構を説明し、その後ヴァイオリン (擦弦)、フルートの例を同一のモデルで説明している。そこでこの論文を中心に、最近になって開発された効率的なシミュレーション技法を展望する。

彼らのモデルは図 1.4にあるように、非線型部分と線形部分とフィードバックループから構成される。同じような構成についての記述が、Adrien と Rodet[AR85]、Rodet[Rod92]、Flechter[Fle92] に見ることができる。クラリネットを例にしたモデルの概要は、次のようである。

リードの運動を無視し、リードとマウスピースとの隙間を単位時間あたりに流れる空気の体積 f は、各時点でのその隙間における圧力降下のみに依存するという事を仮定する。つまりマウ

スピース内の変動する圧力を q、与えられた奏者の口内圧力 p とすれば、f は $p-q$ のみの関数になる。表記する上で便利なように、単に $F(q)$ と書く。関数 $F(q)$ の概形は図 1.5(a) の様である[7]。この図において、傾きが負の部分、つまりマウスピース内の圧力 q が口内圧力 p から少し低くなる部分では、圧力差が増加 (つまり q が減少する) につれて f も増加する、いわゆる抵抗領域である。一方、傾きが正の部分では逆に負性抵抗領域である。また、q_c より q が小さい部分はリードが完全に閉じた状態を示している。

線形な部分を扱う上で重要な考え方は、マウスピース内の圧力の変動をその外側へ出て行く波 $q_o(t)$ と内側へ入ってくる波 $q_i(t)$ の和として表現する事である。断面積が一様の管にそった距離を x、C を音速とすると、圧力は、

$$q(x,t) = q_o(t - x/C) + q_i(t + x/C)$$

これに対応する空気の流れる体積は、

$$Z^{-1}q_o(t - x/C), -Z^{-1}q_o(t - x/C)$$

であって、Z は断面積が一様の管の特性インピーダンスである。

ここで、リード近く $x = 0$ のマウスピース内の圧力の時間変化は、

$$q(t) = q_o(t) + q_i(t) \tag{1.33}$$

リードとマウスピースの隙間を流れる空気の体積変化は、

$$Zf(t) = q_o(t) - q_i(t) \tag{1.34}$$

と書く事ができる。

リード近くで非常に幅の狭いパルス、例えばディラック (Dirac) のデルタ関数 $\delta(t)$、が発生してマウスピースの外へ進むと仮定する。つまり $q_o(t) = \delta(t)$ とする。ところで、管の指穴やベルでは十分低い周波数に対してのみ十分反射させるので、先程のパルスが反射して元のところに戻ってきた時には滑らかな形になる。その反射してきた形を、$r(t)$ と表すことにする 1.5(d)。つまり、$q_o(t) = \delta(t)$ の時に、$q_i(t) = r(t)$ と考える。ここで考慮しているのは管、すなはち線形の部分であるので、重ね合わせの原理が使えて、

$$q_i(t) = r(t) * q_o(t)$$

ここで、アルタリスク "*" は畳込み

$$r(t) * q_o(t) = q_o(t) * r(t) = \int_0^\infty r(t')q_o(t - t')dt'$$

を表す。

図 1.4のモデルによる振動シミュレーションは次の式を解く事に要約される。

非線型な部分

$$f = F(q) \tag{1.35}$$

線形で受動的な部分

$$q(t) = q_h(t) + Zf(t) \tag{1.36}$$

$$q_h(t) = 2q_i(t) = r(t) * q(t) + Zf(t). \tag{1.37}$$

このモデル化とシミュレーションを実行する際の技術的に重要な事をまとめると次のようである。

[7]アンブシュアによって関数全体の垂直方向の倍率が変化する

- 線形部分を一次元の波を左と右への進行波の合成によって表現し、しかも波の往復による減衰を低次のデジタルフィルターを通す事による実現
  この様なモデルは、1979 年に McIntyre と Woodhouse[MW79] が擦弦運動に対して初めて使用した [Smi86]。彼らはまず擦弦運動を次のようなボルテラ (Volterra) 型の非線形積分方程式で表現した。

$$v(t) = \int_0^\infty g(t')f(v(t-t'))dt' \tag{1.38}$$

  ここに $v(t)$ は弓の位置での弦の表面の速度、$g(t)$ はグリーン (Green) 関数あるいはインパルス応答、$f(v)$ は図 1.5(b)[8]の様な形をした、弓と弦との相対速度と摩擦力の非線型関数を表す。この式は、図 1.4を表したものと見る事ができる。
  線形部分のインパルス応答 $g(t)$ は一般に長い時間に対して非ゼロであるため、そのままでは多くの演算時間と記憶容量が必要となる。そこで、インパルス応答を逐次計算するためのモデル化を行うことで、実質的に $g(t)$ の畳込みを効率良く計算することを可能にした。このモデル化というのが、左右それぞれの進行波が往復する様子を遅延回路と低次のデジタルフィルターでの表現である。
- リードや弓などのように楽器を制御するエネルギーを音に変換する非線形な部分の解法
  非線型関数を含む式 1.35, 1.36 のグラフによる解法は、Friedlander と Keller によって初めて用いられた。特に擦弦モデルでは非線型関数 $F(q)$ は素性の悪い、すなはち 3 つの交点をもつ場合がある形をしている。この解のあいまいさを解消するために、

  1. システムは 3 つの交点のうちの中点に対応する状態は決してとらない。
  2. システムは可能な限り現在の交点連続的に移動する。

  という原理を導入した [MSW83]。

Adrien と Rodet[AR85] も、同様のアプローチを発表した。線形部分のモデルとして、例えば弦の場合、質点とバネの直列接続がある。このモデルは Cadoz[CLF84] らが採用しているが、多くの質点の運動を単位時間毎に計算しなければならない。たとえその系の駆動点の運動のみが必要であっても、すべての質点の運動を計算しなければならないので、効率的とは言いがたい。そこで、それぞれ逆向きに進む波を保持する二重の遅延回路で表現し、それを共振 (resonator) モジュールと呼び、励起 (exciter) モジュールと組み合わせてシミュレーション系を構成する方法を提案した。共振モジュールは、小数以下を持つ遅延量を実現し、また弦の剛性をシミュレートするためのオールパスフィルターをそれぞれ持つ二重の遅延回路が反射係数によって結合された、一組の入 / 出力を持つモジュールである。一方、励起モジュールは二組の入 / 出力をもつモジュールで、入力から出力を計算する機能を持つ。

その後、弦の振動を三次元に拡張するための、モジュールの改良、拡張、一般化が行われ [ACR87]、クラリネットにも適用された [ACD88]。

Karplus-Strong アルゴリズム

Kerplus と Strong は、撥弦音とドラム音の合成方法を提案した [KS83]。Kerplus–Strong アルゴリズムとそれにまつわる技術の発展について Simth は、[Smi91b, p.9]

> 「Karplus-Strong アルゴリズムは抽象アルゴリズムとしてリスト[9]されているのは、
> それはテーブルを時間とともに変更する手段を備えた波形テーブル技術だと考えられ

[8] q を v に読み換えていただきたい

[9] 表 1.1を参照

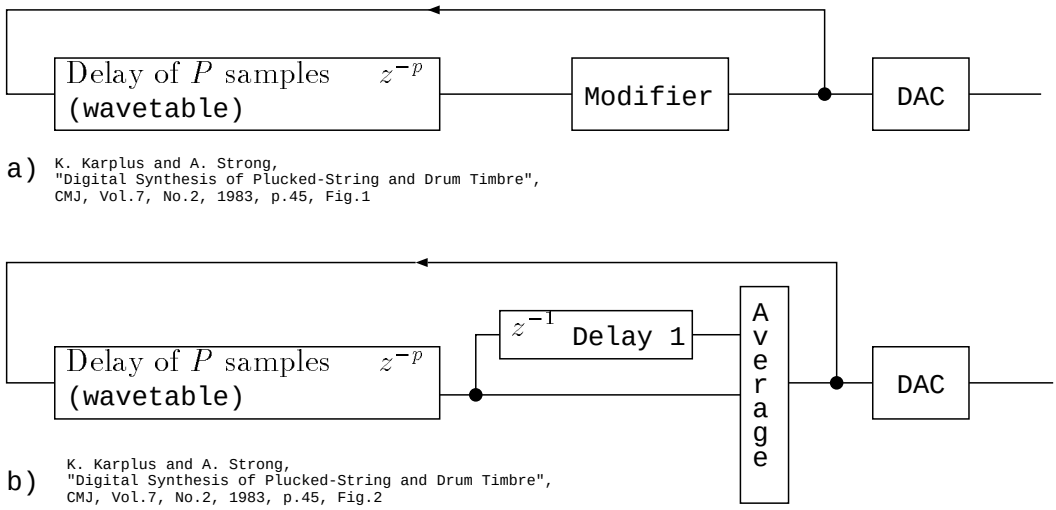


図 1.6: Karplus-Strong アルゴリズム

たからだ。その後それは、McIntyre, Woodhouse それに Scumacher らによった弦の物理モデルの特別な場合であると認識され、それは音楽用に拡張されることになった。Cremer の “method of rounded corner” 「かどを丸める手法」[Cre84] は McIntyre や Woodhouse より先んじていたようだ。全ての人を驚かせた Karplus-Strong アルゴリズムが示した事は「かどを丸める関数」は乗算無しの 2 点平均に単純化できて、しかも音楽的に有効な結果をもたらしうるという事だ。ウェーブガイド合成は、かどを丸める手法の計算上の単純さ保持したままで正確な物理モデルという方向への拡張の一部分である。それは、弦や管のような一次元のウェーブガイドを最も効率的にモデル化するが、Cordis-Anima や Mosaic (ACROE 1990、 Smith 1991) のより一般的な物理モデルへの厳密な方向へ繋げる事も可能である。」

と記述している。

図 1.6 *a)* は、波形テーブル更新による合成方法として、一般化したものである。この方向での研究は、最近セルオートマトンの考え方を適用したものの発表がある [Cha90]。

一方図 1.6 *b)* は、撥弦音の合成アルゴリズムを示している。動作を簡単に説明すると、図中の (wavetable) とある波形メモリにまず、初期波形をロードし、その後二のサンプル点の平均を波形メモリに戻す。その時、波形テーブルの値は 1 サンプル点シフトされる。この操作を繰り返すことによって、撥弦らしい音を合成する。ドラム音の合成の場合には、初期波形をロードする時にランダムノイズバーストを使用し、2 サンプル平均を計算する部分に 1 ビットのランダム波形を加えるといった若干の変更が施される。

[KS83, p.50, Analysis of the Plucked-String Algorithm]

「擦弦モデルの解析

何が、この単純な撥弦アルゴリズムの音をこれほど本物らしくさせているのか? 発音はどのくらい持続するかをどの様にして予測できるのか? これら答えるには、倍音 (overtone) の減衰について調べる必要がある。このアルゴリズムの出力を聞いてみると明らかに、高次倍音は早く減衰するが、基音や低次倍音は長く続いている。低い音程の音は高いものより長く続く。

我々の合成技術の理論的分析と現在あるギター、リュート、マンドリンや他の弦楽器の分析と比較する事は興味ある。残念なことに、発表された分析を見つけられなかっ

たし、我々独自の分析を実行する道具も持っていなかった。いくつかの先行する研究は弦の音を合成するのに物理モデルを使用しているが [HR71a, HR71b]、それらのモデルは我々の技術による高品質な音についての説明に役立たない。」

その中で、デジタルフィルターの設計手法から技法を借用し、減衰時間の推定式を得ている。

同じコンピューター・ミュージック・ジャーナルに、Karplus-Strong アルゴリズムを McIntyre と Woodhouse らの研究を基に拡張している [JS83][10]。ここでの技術は [Smi83] の中でより詳細に述べられているが、弦あるいは管の効率的で物理的な意味を保持したままのシミュレーションの基盤を与えている。たとえばオールパスフィルターによって剛性をもつ弦のふるまいを表現するなど[11]。

Karjakainen 等は、弦楽器と管楽器の物理モデルもどきを汎用浮動小数点 DSP (TMS320C30) で実現した [KLKV91]。彼らは、可変長遅延回路を実現するのにオールパスフィルターの代りにラグランジュ (Lagrange) 補間を利用した。ラグランジュ補間は Sullivan[Sul90] が使用した 2 点補間を多点に一般化したものである。又彼らは双方向の遅延回路を採用しており、この点では Waveguide に近い実装方法と見ることもできよう。

Waveguide

Smith によるウェーブガイド・フィルター (以下 WGF と書く) は、もともと無損失の構成ブロックを使用して残響装置を作る目的で導かれた [Smi87]。Smith は残響装置をそのようにして構成する必要性を以下のよう挙げている [Smi85] 。

1. 拡散接点 (scattering junction) に蓄えられたエネルギーを変化させずに時変動できる。
2. 無損失の回路網の構成方法が得られ、どのような分岐状態にも適合させられる (実時間で変化できる)。
3. リミットサイクルやオーバーフロー発振を相互接続を気にせず、容易に除ける。
4. その構造に中の全ての信号に対し正確な物理的解釈が存在する。
5. 実装における計算効率が良い。

図 1.7に WGF の一部が示されている。WGF は音声の研究分野で声道のモデルの一つとして Kelly と Lochbaum により提案された進行波による表現 [Fla83] をベースにデジタル・フィルターとしての位置付けを行なったものと考えられる。隣接する音響管の接続点のでの音波の透過 / 反射を表現している 4 つの乗算を持つ回路[12] は 1 つの乗算の回路 [GM76] で実現できるが、長い音響管を直接このようなモデルで表現するとかなりの計算量を必要とする。したがって、分散している接続点と遅延を集中させることが実用上行われる事が多い。遅延ループによる線形部分のは、質量とバネの系を記述する差分方程式の数値的に積分する有限要素のアプローチに比べ非常に効率的であるので、その後、楽器の実時間シミュレーションに広く応用されるようになった [Smi86]。

CORDIS-ANIMA

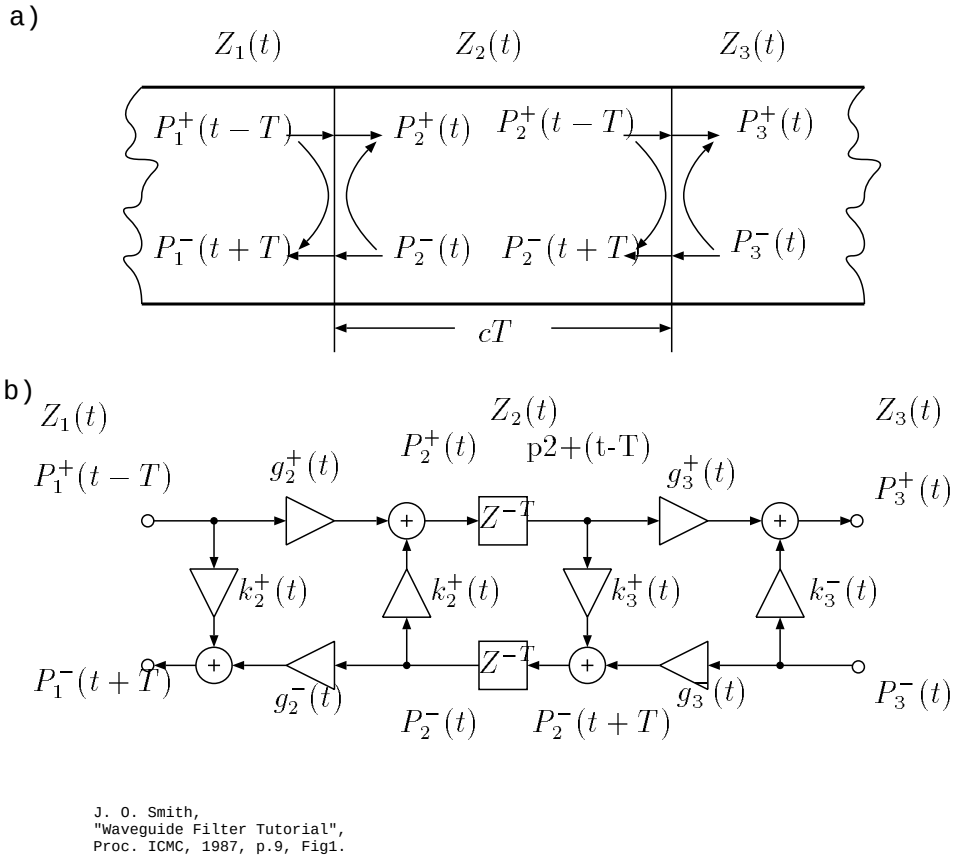
Cadoz らは、1978 年ころから楽器の本質の考察から出発し、新しいテクノロジーによる楽器の基本的な研究を続けている。コンピューターミュージックジャーナルには 1984 年[13]に掲載

[10] この論文で著者は [MW79] を 1960 年としている

[11] このアイディアは Allen によるものらしい [JS83]

[12] Kelly–Lochbaum junction と呼んでいる

[13] 論文が書かれたのは、1980 年。1982 年には Cordis-Temps-Reel (CTR) という実時間信号処理装置を開発し、この論文にあるシミュレーションアルゴリズムによる音を実時間で合成した



J. O. Smith,
"Waveguide Filter Tutorial",
Proc. ICMC, 1987, p.9, Fig1.

図 1.7: ウェーブガイド・デジタル・フィルター

されたのが英語の論文 (C. Roads、 F. Chadabe 訳) としては最も初期のもののようである [CLF84]。

コンピューター音楽が音楽家に提供して来た、伝統的な自然楽器に基づいた音楽的手法とは違う手法、つまり楽音合成あるいは音響操作手段は、本当に音楽家にとって十分であったかどうか? このような疑問を提示し、それに対する回答を与える目的の研究が、フランスの人工知能基礎情報研究所 L.I.F.I.A.[14] の表現具開発研究会 A.C.R.O.E. [15]で、 Cadoz を中心に初められた。

CORDIS-ANIMA という、 CORDIS システムに視覚情報 (アニーメーション) を付加したシステム [FRLC86]、ジェスチャの記述を基本とした作曲について [Cad88]、ジェスチャの力学的モデル [GF88]、いろいろな楽器のシミュレーションを行うために規格化したモデル構築方法 [CF90]、ジェスチャの記述 [CR90]、力学的フィードバック装置 [CLF90a, CLF90b] など一連の研究を行っている。特に Gibet と Florens は楽器 – 演奏者の連成系について詳しく述べている [GF88]。

モーダル合成

モーダル・モデルは質点、バネ、摩擦力を組みにした要素によって構成される [CF90]。 MOSAIC [16]は物理モデルを基にした楽音合成用プログラムである。この中で使用されるモーダル合

[14]Laboratoire d'Informatique Fondamentale et d'Inteligence Artificielle

[15]Association pour la Création la Recherche sur les Outils d'Expression

[16]Modal Synthesis and Analysis with Interactive Control

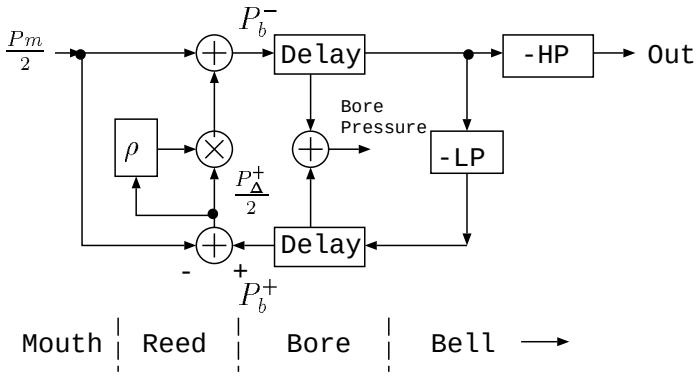


図 1.8: クラリネット型のモデル

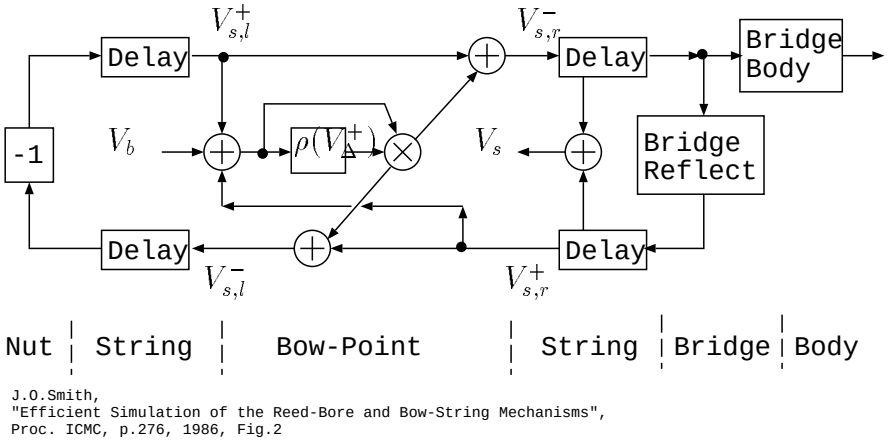


図 1.9: 擦弦 (バイオリン) のモデル

成 [ORA90] による物理モデルは、Adrien によって作成された。MOSAIC はスキーマ (Scheme) プログラム言語用のインタープリターを基に作成された。インタープリターに関する記述は、[MA91] に見ることができる。

実現例

- クラリネット
  図 1.8は Smith[Smi86] によるクラリネットの基本的モデルによる合成回路のブロック図である。
  また、[MLC90] や 3– ポート拡散接続によって指穴を表現したもの [HCS91] の, あるいはベルの部分をより精密にモデル化した例 [Smi91a] などの研究がなされている。
- バイオリン
  図 1.9 は Smith[Smi86] によるバイオリンの基本的モデルによる合成回路のブロック図である。 Malouf、Lentczner、Chafe らも同様なモデルを使用している [MLC90]。
  Adrien らは [AR85, ACR87, ACD88] 3 次元的運動を考慮した弓や駒のより精密なモデルを提案した。

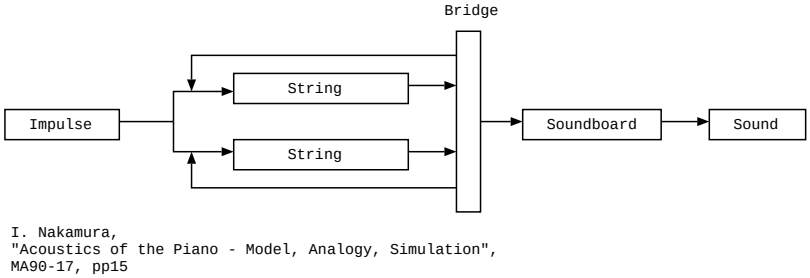


図 1.10: ピアノ音合成のモデル (中村による)

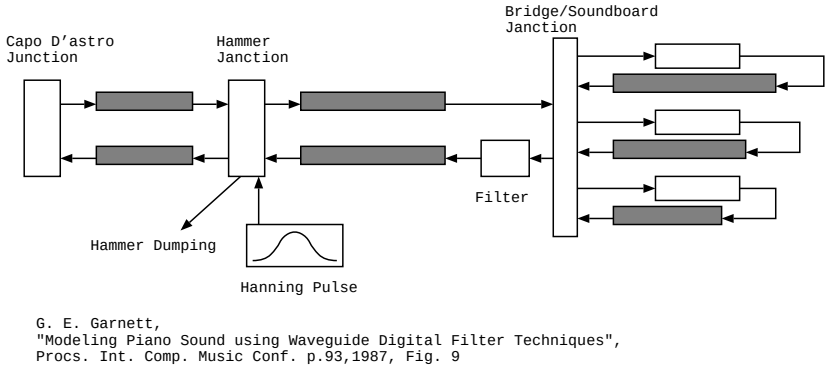


図 1.11: Wageguide によるピアノのモデル (G. Garnett による)

ヴァイオリンの研究は [近藤 90]、海外のものは [Huc83, カリ 83, Cre84] 等に、日本のものは [近藤 90] によって詳しく紹介されている。例えば、岸、藤尾らは [岸 憲 86] 等価回路的考察に基づいて、バイオリン属楽器の擦弦振動をシミュレートする方法を提案した。すなはち、弓を定速度源、弦を擦弦点から見た機械インピーダンスで表し、両者を摩擦特性を表す非線型の抵抗を介して結合することにより擦弦振動系全体の等価回路を得、この回路の動作をデジタル計算機によりシミュレートする方法を示した。得られた非線型連立微分方程式をルンゲ・クッタ・ギル (Runge-Kutta-Gill) 法によって解いている。

- ピアノ
  中村はピアノの一連の研究の中で、電気回路網によるアナロジーによって二線条分布定数回路に置き換え、アナログ回路によるシミュレーション、差分近似による直接シミュレーション、デジタルフィルターによるシミュレーションなど行っており、[中村 90a] に簡潔にまとめられている。図 1.10は、デジタルフィルターによるシミュレーションに採用した二本弦の場合のモデルを示している。弦はそのインパルス応答を直接表現するための FIR[17] フィルターによって、響板は二次の IIR[18] フィルターによって近似されている [NI88]。
  Garnett による Waveguide を使用した例を図 1.11に示す。[Gar87] 打弦点を表す接続 (junction) 点の反射係数を時変動させることでハンマーと弦の接触、非接触の状態を表現している。またハンマーによる励起をハニング (Hanning) 型のパルス

[17]Finite Inpulse Response

[18]Infinite Inpulse Response

$$f = 0.5 + 0.5\cos(\frac{2\pi n}{t_h})$$

で近似し、そのパルスの幅を打鍵速度が早い時は狭く、遅い時は広くする制御を行っている。

例えばSuzukiによるハンマーと弦の相互作用[Suz88]を考慮した同様のモデルはBorin[BDS89]が詳細に述べている。

- その他
  - ドラム – Karplus[KS83]
  - フルート – Karjakaninen[KLKV91]
  - トランペット – Cook[Coo91]、DepalleとRodet[DR92]
  - アコースティックギター – Karjakainen[KLKV91]
  - 声 – Cook[Coo89]
  - エレクトリックギター – Sullivan[Sul90]

演奏者

物理モデルの合成系をコントロールする事は、主に実際の演奏家がする様な方法でそのパラメータに影響を与える問題だ。それには、

1. 実際の演奏者に対する双方向の物理的結合装置(インターフェース)の構築。
2. コンピューター上に演奏者のジェスチャと等価なものの生成。
   - ジェスチャのサンプリング
   - 演奏者の物理モデル

の二のアプローチがある。

演奏者とのインターフェース

Cadozらは、力学的フィードバックを備えた演奏者とのインターフェースを幾つか試作している。[CLF84]によれば、

> 「**最初のジェスチャ変換器**
>
> この装置はJ. L. Florensによって1978に作られた。それは、二つの水平なガイドの上を動く移動装置に取り付けられた“棒”から成っている。
>
> **二番目のジェスチャ変換器**
>
> これは1980年に考案され実現されたもので、ピアノの鍵盤からひらめいたユニークな鍵、動きが約5cmと大きいが、からできている。この装置の発想は、最初のものに比べ機構的により密で、新しいモーターとセンサー部品を使用し、他の装置とは全く違う動作を可能としている。定常的に回転するモーターによる振動や騒音は無い。それはより良い応答時間を持ち、装置の空間的精度は十数ミクロンのオーダである。
>
> しかし、モーターによる力はあまり強くないので、シミュレーションの限界がある。返しの動作を見た場合、シミュレーションの最も困難な事は、完全に動かない障害物である。加えられる力の大きさは、非常に短い時間で大変大きく成ってしまう。この性質を軽量で小型の限られた機構系でモデル化するのは困難である。
>
> 最初のシステムと反対に、このシステムは二つの相補的なモードで使用できる。

- 位置を返す力センサーとして
- 力を返位置センサーとして

操作する人から見れば、この二つの状況はサーボ制御が組み込まれていれば等価である。」

1990 年には触覚フィードバック付のキーボード型インターフェースを発表している。[CLF90a, CLF90b]。この装置は鍵一つが *slice motor* と呼んでいる薄い駆動装置とセンサーから成るモジュールで、それらをちょうど食パンをならべるように接続して一つの鍵盤を構成している。モーターは、可動コイルが直線的に 15mm 動くようになっており、厚さ 13.75mm、重さ 200g、最大連続出力 40N、瞬間最大出力 80N である。初めの第二番目の装置が、鍵をモデルとして作成されたが、強力な力学フィードバックが可能である代りに大きく重いため、鍵盤を構成するには不適当であった。新しいタイプのものは、小型で軽量であるが、小さな力フィードバックになった。もちろん、これは鍵盤として使用されるが、それ以外に取手のついたコの字型のアダプターの両端をそれぞれ別の鍵に取り付け、二次元インターフェースを構成した。

その他、グランドピアノのアクションのシミュレーションについて研究がある [Gil92]。

ジェスチャのサンプリング

Chafe はヴァイオリンのモデルをいわゆるルールベースにより駆動するシステムを開発した [Cha85]。演奏家の手は、手らしく動くように拘束されたプログラマブルな物として表現される。音楽イベントはコントロールソフトウェアによって、手のメカニカルな動きへと翻訳される。ルールは実際の演奏から得たのではなく、試行錯誤によって得たものである。

演奏者の物理モデル

Gibet らは、演奏者の腕の物理的モデルを示し、ドラムを叩く動作とドラムとの相互作用をシミュレーションした。[GM90]。その中で、腕は 4 自由度を持つ 3 つの副振り子としてモデル化され、適応型制御される。ドラムは弾性膜として二次元波動方程式によって記述されている。そして、それぞれの系は、接触しない間はそれぞれ独立の微分方程式によって記述され、接触した時には作用反作用の原理に従う。

1.3.3 室内音場 – 聴取者系

室内音場及び聴取者を考慮に入れた音場の研究は、建築音響、電気音響等の分野でさかんに研究されている。最近は、幾何音響理論に基づく音線シミュレーションやベルジェロン (Bergeron) 法と呼ばれる波動伝播解析手法等が実用化されつつある。音線シミュレーション方法は波動方程式を差分法によってシミュレーションする場合とは逆に波動性の無視できる周波数の高い領域に適用できる。一方、周波数の低い領域での特性を波動的に取り扱う場合の問題点として、室内音場の解析的表現が十分でないため、数値解析のアプローチが重要である。

さて、コンピューター音楽におけるこの領域の研究は、Kendall らによる研究を中心に概観するのが適当と思われる。彼らはノースウエタン大学で 1984 年に聴取空間の音響特性の総合的なシミュレーションシステムを構築した [KM84]。そのコントロールソフトウェアの解説 [KMF+86] の中で、

「もともとの Schroeder の残響装置が “残響時間” という一つのパラメータしか持っていなかったのとは対照的に、“空間的残響” はシミュレーションされる環境の音響

特性の詳細な仕様を要求する。また、発音体と聴取者のダイナミックな関係 (Chowning [Cho71] と Moore [Moo83] によって開発された空間処理装置によるそれぞれの要求事項) も必要である。詳細を決定する過程は、物理的な状況設定のモデルを基本にする事によって補助さる。ユーザーは動的に部屋の大きさ、音源の位置、聴取者の位置などのパラメータを操作できるが、その他の壁の吸音係数などのパラメータは静的のままである。空間的残響を制御する事は、結局は空間的印象の主観的次元を心理学的に決定する方向に向かうのだが、物理的モデルは創造的潜在能力の探究や心理学的次元の基本的構成の研究に対する出発点を与える。」

と記述している。

このシステムでは、11 個のパラメータ

部屋の大きさ	幅、高さ、奥行き	3 個
音源の位置	3 次元座標	3 個
聴取者の位置	3 次元座標	3 個
聴取者の向き	方位角、仰角	2 個

を各離散時間毎に指定できる。あるいは、ある特定な時点のみを指定し、Bezier 曲線によって補間する事もできる。

その後、コンピューター音楽におけるこの分野の展望を述べている [Ken91]。最近では、NeXT ワークステーション上に PinnaWorks という音像制御システムを構築している。PinnaWorks は Kemer のマネキン (ダミー・ヘッド) によって得られた HRTF[19] を使用し、Schroeder-Atal のクロストークキャンセル (図 1.12) を含むこともできる。さらに Polhemus の 6 自由度位置センサーによる直接的な音像制御が可能である [Ken90]。

HRTF によらない音像定位も使われているが、マン・マシンインターフェースに重点が置かれた研究といえるだろう。例えば、円上に等間隔に複数のスピーカーを配置し、隣接する 2 つのスピーカーのインテンシティを制御することで、音源の定位 (距離、方位角) を実現している。またドップラー効果を付加するために、音源の音程を制御している [Bos90]。

Garton[Gar90] は野外、例えば狭谷や森での音場をフラクタルを用いてシミュレーションする事をコンピューター・グラフィクスからヒントを得て提案している。そのような音場の反射パターン (インパルス応答) を、あるパターンを再帰的な処理を施して生成する。しかし時々変動させるなどの補助手段を使わないと効果があまり無いようである。

空間音響については [イェ 85]、バイノーラル再生技術については [浜田 92] 等にまとめられている。

[19] Head Related Transfer Function、頭部伝達関数

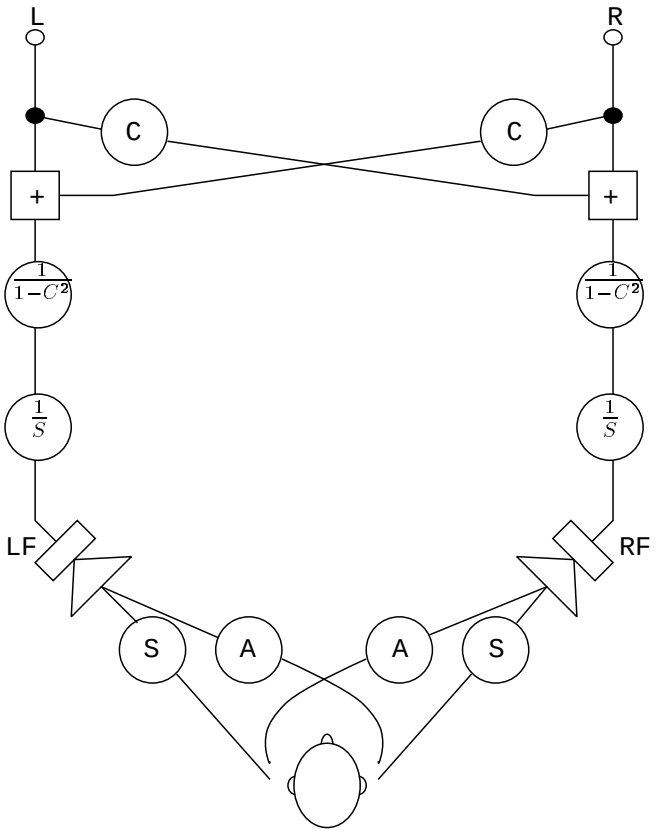


図 1.12: Atal-Schroeder クロストーク キャンセル フィルター

参考文献

[ABE91] 安部素嗣. 楽器音の学習認識システム. 東京大学工学部 計数工学科 卒業論文, 1991.

[ACD88] J. Adrien, R. Causse, and E. Duccasse. Dynamic Modeling of Stringed and Wind Instruments, Sound Synthesis by Physical Models. In *Int. Comp. Music Conf.*, pp. 265–276, 1988.

[ACR87] J. Adrien, R. Causse, and X. Rodet. Sound Synthesis by Physical Models, Application to Strings. In *Int. Comp. Music Conf.*, pp. 264–269, 1987.

[AR85] J. Adrien and X. Rodet. Physical Models of Instruments: A Modular Approach, Application to Strings. In *Int. Comp. Music Conf.*, pp. 85–89, 1985.

[BDS89] G. Borin, G. Depoli, and A. Sarti. A modular approach to excitor–resonator interaction in physical models synthesis. In *Int. Comp. Music Conf.*, pp. 46–50, 1989.

[Ben89] Gerald Bennett. Digital synthesis of the singing voice. In *Proceedings of ISMIS*, pp. 32–36, March 1989.

[BH92] James Beauchamp and Andrew Horner. Extended nonlinear waveshaping analysis/synthesis technique. In *Proceedings of ICMC*, pp. 2–5, 1992.

[BJ78] Boer, E. and Jongh, H. R. On cochlear encoding: potentialities and limitations of the reverse-correlation technique. *J. Acoust. Soc. Am.*, Vol. 63, No. 1, pp. 115–135, 1978.

[Bos90] M. Bosi. An interactive real-time system for the control of sound localization. In *Int. Comp. Music Conf.*, pp. 112–114, 1990.

[Bre90] Bregman, A. S. *Auditory Scene Analysis.* The MIT Press, 1990.

[BS91] E. Byyan and Mark J.T. Smith. An analysis–by–synthesis approach to sinusoidal modeling applied to the analysis and synthesis of musical tones. In *Proceedings of 1991 ICMC*, pp. 356–359, 1991.

[Cad88] C. Cadoz. Instrumental Gesture and Musical Compostion. In *Int. Comp. Music Conf.*, pp. 1–12, 1988.

[CF90] C. Cadoz and J. Florens. Modular modelisation and simulation of the instrument. In *Int. Comp. Music Conf.*, pp. 75–78, 1990.

[Cha85] C. Chafe. Control of Bowed String Synthesis from a Physical Model. Technical Report CCRMA Tech. Rep. STAN–M–32, Stanford University, 1985.

[Cha90] J. Chareyron. Digital synthesis of self-modifying waveforms by means of linear automata. *Computer Music Journal*, Vol. 14, No. 4, pp. 25–41, 1990.

[Cho71] J. M. Chowning. The Simulation of Moving Sound Sources. *J. Audio Eng. Soc.*, Vol. 19, No. ??, pp. 2–6, 1971.

[Cho73] John Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of AES*, Vol. 21, No. 7, pp. 526–534, 1973.

[CLF84] C. Cadoz, A. Luciani, and J. Florens. Responsive Input Devices and Sound Synthesis by Simulation of Instrumental Mechanisms: The Cordis System. *Computer Music Journal*, Vol. 8, No. 3, pp. 60–73, 1984.

[CLF90a] C. Cadoz, L. Lisowski, and J. Florens. Modular Feedback Keyboad. In *Int. Comp. Music Conf.*, pp. 379–382, 1990.

[CLF90b] C. Cadoz, L. Lisowski, and J. Florens. Modular Feedback Keyboard Design. *Computer Music Journal*, Vol. 14, No. 2, pp. 47–51, 1990.

[CM83] Claasen, T. A. C. M. and Mecklenbrauker, W. F. G. The aliasing problem in discrete-time Wigner distributions. *IEEE Trans. on ASSP*, Vol. 31, No. 7, pp. 1067–1072, 1983.

[CMBP92] John Michael Clarke, Peter Manning, Ron Berry, and Alan Purvis. Vocel : New implementations of the fof synthesis method. In *Proceedings of 1988 ICMC*, pp. 357–371, 1992.

[Coh66] Cohen, L. Generalized phase-space distribution functions. *Math. Phys.*, Vol. 7, pp. 781–786, 1966.

[Coo89] P. R. Cook. Synthesis of the singing voice using a physically parameterized model of the human vocal tract. In *Int. Comp. Music Conf.*, pp. 69–72, 1989.

[Coo91] P. R. Cook. TBone: An Interactive Waveguide Brass Instrument Synthesis Workbench for the NeXT MAchine. In *Int. Comp. Music Conf.*, pp. 297–299, 1991.

[CR90] C. Cadoz and C. Ramstein. Catpture, Representation, and "Composition" of the Instrumental Gesture. In *Int. Comp. Music Conf.*, pp. 53–56, 1990.

[Cre84] L. Cremer. *The Physics of the Violin*. MIT Press, Cambridge MA, 1984.

[CTD83] Chester, D.,, Taylor, F. J., and Doyle, M. On the Wigner distribution. In *ICASSP 83*, pp. 491–494, 1983.

[CW89] Choi, H. and Williams, W. J. Improved time–frequency representation of multicomponent signals using exponential kernels. *IEEE Trans. on ASSP*, Vol. 37, No. 6, pp. 862–871, 1989.

[Dod89] Charles Dodge. Synthesized voices and musical fractals: Two natural models for computer music. In *Proceedings of ISMIS*, pp. 46–50, March 1989.

[DR92] P. Depalle and X. Rodet. A physical model of lips and trumpet. In *Int. Comp. Music Conf.*, pp. 132–133, 1992.

[FBS62] Fletcher, H.,, Blackman, E. D., and Sratton, R. Quality of piano tones. *J. Acoust. Soc. Am.*, Vol. 34, No. 6, pp. 749–761, 1962.

[Fla83] J. F. Flanagan. *Speach Analysis Synthesis and Perception.* Springer-Verlag, 2nd edition, 1983.

[Fle92] N. H. Fletcher. Linearity and nonlinearity in musical instruments. In *Int. Symposium on Musical Acoustics*, pp. 15–18, 1992.

[FRLC86] J. Florens, A. Razafindrakoto, A Luciani, and C. Cadoz. Optimized Real Time Simulation of Objects for Musical Synthesis and Animated Image Synthesis. In *Int. Comp. Music Conf.*, pp. 65–70, 1986.

[Gar87] G. E. Garnett. Modeling piano sound using waveguide digital filtering techniques. In *Int. Comp. Music Conf.*, pp. 89–95, 1987.

[Gar90] B. Garton. Two new approaches to the simulation of acoustic spaces. In *Int. Comp. Music Conf.*, pp. 115–117, 1990.

[GF88] S. Gibet and J. Florens. Instrumental Gesture Modeling by Identification with Time–Varying Mechanical Models. In *Int. Comp. Music Conf.*, pp. 28–40, 1988.

[Gil92] B. Gillespie. Dynamical modeling of the grand piano action. In *Int. Comp. Music Conf.*, pp. 77–80, 1992.

[GM76] A. H. Gray and J. D. Markel. *Linear Prediction of Speech.* Springer-Verlag, 1976.

[GM90] S. Gibet and P Marteau. Gestural control of sound synthesis. In *Int. Comp. Music Conf.*, pp. 387–391, 1990.

[HAR83] 原, 井口. 複素スペクトルを用いた周波数同定. 計測論, Vol. 19, No. 9, pp. 718–723, 1983.

[HBH92] Andrew Horner, James Beauchamp, and Lippold Haken. Wavetable and FM matching synthesis of musical instruments tones. In *Proceedings of 1992 ICMC*, pp. 18–21, 1992.

[HCS91] S. E. Hirchman, P. R. Cook, and J. O. Smith. Digital waveguide modeling of reed woodwinds: An interactive development. In *Int. Comp. Music Conf.*, pp. 300–303, 1991.

[HH92] Bryan Holloway and Lippold Haken. A sinusoidal synthesis algorithm for generating transitions between notes. In *Proceedings of 1992 ICMC*, pp. 14–17, 1992.

[HR71a] L. Hiller and P. Ruiz. Synthesizing Musical Sounds by Solving the Wage Equation for Vibrating Objects: Part I. *J. Audio Eng. Soc.*, Vol. 19, No. 6, pp. 462–470, 1971.

[HR71b] L. Hiller and P. Ruiz. Synthesizing Musical Sounds by Solving the Wage Equation for Vibrating Objects: Part II. *J. Audio Eng. Soc.*, Vol. 19, No. 7, pp. 542–551, 1971.

[Huc83] C. M. Huchins. A history of violin research. *J. Acoust. Soc. Amer.*, Vol. 73, No. ?, pp. 1421–1449, 1983.

[IRI91] 入野, 河原. 聴覚系インパルス応答を用いた wavelet 変換による分析合成. 音講論集, 1991.

[JS83] D. A. Jaffe and J. O. Smith. Extensions of the Karplus–Strong Plucked–String Algorithm. *Computer Music Journal*, Vol. 7, No. 2, pp. 56–69, 1983.

[Ken90] G. Kendall. Pinnaworks. In *Int. Comp. Music Conf.*, pp. 122–123, 1990.

[Ken91] G. S. Kendall. Visualization by ear: Auditory imagery for scientific visualization and virtual reality. *Computer Music Journal*, Vol. 15, No. 4, pp. 70–73, 1991.

[KKOH92] A. I. Katrami, R. Kirk, R. Orton, and A. Hunt. Deconstructing the phase vocoder. In *Proceedings of 1992 ICMC*, pp. 392–393, 1992.

[KLKV91] M. Karjakainen, U. K. Laine, T. Kaakso, and V. Välimäki. Transmission-Line Modeling and Real-Time Synthesis of String and Wind Instruments. In *Int. Comp. Music Conf.*, pp. 293–296, 1991.

[KM84] G. S. Kendall and W. L. Martens. Simulating the cues of spatial hearing in natural environments. In *Int. Comp. Music Conf.*, pp. 111–125, 1984.

[KMF+86] G. S. Kendall, W. L. Martens, D. J. Freed, M. D. Ludwig, and R. W. Larstens. Spatial procesing software at northwestern computer music. In *Int. Comp. Music Conf.*, pp. 285–292, 1986.

[Kro88] Kronland-Martinet, R. The wavelet transform for analysis, synthesis, and processing of speech and music sounds. *Computer Music J.*, Vol. 12, No. 4, pp. 11–20, 1988.

[KS83] K. Karplus and A. Strong. Digital Synthesis of Plucked–String and Drum Timbers. *Computer Music Journal*, Vol. 7, No. 2, pp. 43–55, 1983.

[KT92] Kashino, K. and Tanaka, H. A sound source separation system using spectral features integrated by the Dempster's law of combination. In *Annual Report of the Engineering Research Institute*, volume 51, pp. 67–72. Faculty of Engineering, University of Tokyo, 1992.

[MA91] J. D. Morrison and J. Adrien. Control Mechanisms in the MOSAIC Synthesis Program. In *Int. Comp. Music Conf.*, pp. 19–22, 1991.

[MH91] Meddis, R. and Hewitt, M. J. Virtual pitch and phase sensitivity of a computer model of the auditory periphery: I. phase sensitivity. *J. Acoust. Soc. Am.*, Vol. 89, No. 6, pp. 2883–2894, 1991.

[MLC90] F. L. Malouf, M. Lentczner, and C. Chafe. A real-time implementation of physical models. In *Int. Comp. Music Conf.*, pp. 191–193, 1990.

[Moo83] F. R. Moore. A General Model for Spatial Processing of Sounds. *Computer Music Journal*, Vol. 7, No. 3, pp. 6–15, 1983.

[Moo90] F. Richard Moore. *Elements of Computer Music*. Prentice Hall, 1990.

[MSW83] M. E. McIntyre, R. T. Schumacher, and J. Woodhouse. On the Oscillations of Musical Instruments. *J. Acoust. Soc. Amer.*, Vol. 74, No. 5, pp. 1325–1345, 1983.

[MW79] M. E. McIntyre and J. Woodhouse. On the Fundamentals of Bowed String Dynamics. *Acustrica*, Vol. 43, No. 2, pp. 93–108, 1979.

[NAG91] 長沼, 中村. プロダクト核を持つ時間周波数表現 – 理論とピアノ音への応用 –. 日本音響学会音楽音響研究会資料, MA91-13 / 音楽情報科学研究会資料, 34-3, 1991.

[Nag92] Yoichi Nagashima. Real-time control system for 'psuedo granulation'. In *Proceedings of 1992 ICMC*, pp. 404–405, 1992.

[NI88] I. Nakamura and S. Iwaoka. Piano tone synthesis using digital filters by computer simulation. In *Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pp. 1293–1296, 1988.

[OHK91] Richard Orton, Andy Hunt, and Ross Kirk. Graphical control of granular synthesis using cellular autumata and freehand program. In *Proceedings of 1991 ICMC*, pp. 416–418, 1991.

[ORA90] C. Olivier, L. Ronan, and J. Adrien. Modal synthesis compilation of mechanical sub–structures and acoustical sub–system. In *Int. Comp. Music Conf.*, pp. 57–59, 1990.

[Ril87] Riley, M. D. Beyond quasi-stationarity: designing time–frequency representations for speech signals. In *ICASSP 87*, pp. 657–660, 1987.

[Ris89] Jean-Claude Risset. Issues for new music: Bridging gaps – between computer sound and instrumental sound; between real-time and delayed synthesis; between digital synthesis and digital processing of natural sound. In *Proceedings of ISMIS*, pp. 106–110, March 1989.

[Roa78] Curtis Roads. Granular synthesis of sound. *CMJ*, Vol. 2, No. 2, pp. 61–62, 1978.

[Rod89] Xavier Rodet. Digital signal synthesis in music: Methods, means and applications. In *Proceedings of ISMIS*, pp. 91–95, March 1989.

[Rod92] X. Rodet. Nolinear oscillator models of musical instrument excitation. In *Int. Comp. Music Conf.*, pp. 412–413, 1992.

[Ros92] Dave Rossum. Making digital filters sound "analog". In *Proceedings of 1992 ICMC*, pp. 30–33, 1992.

[Smi83] J. O. Smith. Techniques for Digital Filter Design and System Identification with Appplication to the Violin. Technical Report CCRMA Tech. Rep. STAN–M–14, Stanford University, 1983.

[Smi85] J. O. Smith. A New Approach to Digital Reverbration using Closed Waveguide Networks. In *Int. Comp. Music Conf.*, pp. 47–53, 1985.

[Smi86] J. O. Smith. Efficient Simulation of the Reed–Bore and Bow–String Mechanisms. In *Int. Comp. Music Conf.*, pp. 275–279, 1986.

[Smi87] J. O. Smith. Waveguide Filter Tutorial. In *Int. Comp. Music Conf.*, pp. 9–16, 1987.

[Smi91a] J. O. Smith. Wageguide Simulation of Non–Cylindrical Acoustic Tubes. In *Int. Comp. Music Conf.*, pp. 304–307, 1991.

[Smi91b] Julius Orion Smith. Viewpoints on the history of digital synthesis. In *Proceedings of 1991 ICMC*, pp. 1–10, 1991.

[SRD88] Marie-Helene Serra, Dean Rubine, and Roger B. Dannenberg. The analysis and resynthesis of tones via spectral interpolation. In *Proceedings of 1988 ICMC*, pp. 322–332, 1988.

[SS89] Xavier Serra and Julius O. Smith. Spectral modeling synthesis. In *Proceedings of 1989 ICMC*, pp. 281–284, 1989.

[Str89] John M. Strawn. Analysis and synthesis of musical transitions using the discrete short-time fourier transform. *Journal of AES*, Vol. 35, No. 1/2, pp. 3–13, 1989.

[Sul90] C. R. Sullivan. Extending the Karplus–Strong Algorithm to Synthesize Electic Guitar Timbres with Distortion and Feedback. *Computer Music Journal*, Vol. 14, No. 3, pp. 26–37, 1990.

[Suz88] H. Suzuki. Model analysis of a hammer-string interaction. In *Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pp. 1285–1288, 1988.

[TAK92] 高澤嘉光. 周波数分離法による鐘の振動解析. 信学技報 EA91-100, 1992.

[UEN92] 上野山, 小沢. ウェーブレット変換による打楽器音の分析・合成. 数理科学, Vol. 30, No. 12, pp. 31–35, 1992.

[VK91] Moore, R. K., Varga, A. P. and Kadirkamanatha, M. Automatic separation of speech and other complex sounds using hidden Markov model decomposition. In *Institute of Acoustics Speech Group Meeting*. Sussex University, 1991.

[Wec90] Wechsler, H. *Computational Vision*. Academic Press, 1990.

[YAM77] 山口, 安藤. 短時間スペクトル分析法の楽器音への適用. 音響誌, Vol. 33, No. 6,, 1977.

[イェ 85] イェンス ブラウエルト (森本政之、後藤敏幸 訳). 空間音響. 鹿島出版会, 1985.

[カリ 83] カリーン M. ハッチンス (中村 勲、永井洋平 訳). バイオリンの音響学. 日本音響学会誌, Vol. 39, No. 3, pp. 204–213, 1983.

[安藤 89] 安藤由典, 大串健吾, 井戸川徹, 中村勲, 岩宮眞一郎. 第 13 回国際音響会議 (ベオグラード) 音楽音響部門報告. 音楽音響研究会資料, Vol. 8, No. 4, pp. 27–32, 1989.

[岸 憲 86] 岸 憲史, 藤尾宗庸. 擦弦振動のシミュレーション. 音楽音響研究会資料, Vol. 5, No. 2, pp. 9–16, 1986.

[岩宮 86] 岩宮眞一郎, 井戸川徹, 安藤由典. 第 12 回国際音響学会議 (トロント)– 音楽音響部門 –. 音楽音響研究会資料, Vol. 5, No. 5, pp. 5–9, 1986.

[久保 86] 久保田 秀美, 白砂 昭一, 近藤正夫. 国際音楽音響シンポジウム報告. 音楽音響研究会資料, Vol. 5, No. 5, pp. 1–4, 1986.

[久保 91] 久保田秀美, 近藤正夫. アメリカ音響学会第 121 回年会参加、およびキャットガット音楽音響研究会参加報告. 音楽音響研究会資料, Vol. 10, No. 4, pp. 23–27, 1991.

[近藤 90] 近藤正夫, 岸 憲史, 小嶋 卓, 久保田秀美. 日本におけるバイオリンの研究. 日本音響学会誌, Vol. 46, No. 10, pp. 854–863, 1990.

[中村 86] 中村勲. 1986 年音響・音声・信号処理国際会議の “音楽音響における信号処理” について. 音楽音響研究会資料, Vol. 5, No. 2, pp. 1–2, 1986.

[中村 89a] 中村勲, 矢野浩之, 井戸川徹, 安藤由典, 近藤正夫. 音楽音響国際シンポジウム (ISMA89) 報告. 音楽音響研究会資料, Vol. 8, No. 4, pp. 15–26, 1989.

[中村 89b] 中村勲, 鈴木英男, 井戸川徹, 高橋準二, 田口友康, 中村俊一. 日米音響学会第 2 回ジョイント・ミーティング音楽音響関係報告. 音楽音響研究会資料, Vol. 7, No. 6, pp. 30–36, 1989.

[中村 90a] 中村勲. ピアノの音響学 – モデル、アナロジー、シミュレーション –. 音楽音響研究会資料, Vol. 9, No. 5, pp. 15–22, 1990.

[中村 90b] 中村勲. 音楽音響研究 10 年間の動向. 音楽音響研究会資料, Vol. 9, No. 5, pp. 1–2, 1990.

[藤崎 89] 藤崎博也, 中村勲, 安藤由典, 田口友康. 音楽と情報科学国際シンポジウム報告. 音楽音響研究会資料, Vol. 8, No. 2, pp. 15–18, 1989.

[浜田 92] 浜田晴夫. バイノーラル音場再生系について. 日本音響学会誌, Vol. 48, No. 4, pp. 250–213, 1992.

第 2 章

音楽情報の表現

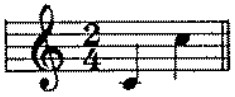
2.1 楽譜表記方式 (松島)

音楽情報を計算機で処理するには, 元になる音楽情報を計算機内に取り込む必要がある. この仕事は, 従来人手により行われることが多く, そのために音楽の計算機可読表現が考案されてきた [松島 88] [BM70][Bri86][Bro70] [Fox87][GL70][Rob67][Wen72]. この内, 比較的成功したものに DARMS[Eri75][Eri76], MUSTRAN[Wen70], MUSICA[DdP82][DP76] などがあり, 音楽解析, 楽譜印刷, 演奏など特定の研究領域では一応の成果を収めてはいるものの, いまだ音楽の表記法に関して支配的なものはない. 演奏情報に限っていえば, 現在市販のほとんどの電子楽器が採用している MIDI 規格がある. この規格で定められている音楽情報の表記法は, 例えばピアノのような楽器を演奏するとき, あるいは演奏したときの鍵盤の動きを記述するような表記法である. 即ち, 鍵盤の押された時刻と離された時刻で指定するものであり, 具体的な演奏と一対一に対応している. これは, 楽譜のような抽象化された情報に比べて低水準の表記法であり, 楽譜情報については考慮されていない. 一方, 種々多様な音楽情報処理システムが開発されるにつれて, 音楽情報の標準的な記述方式の必要性が増大している. 最近になって, ANSI で音楽表記の標準化案の作成を行っている.

ここでは計算機による音楽演奏を目的とした計算機用の音楽表記法にはあまり触れず, 楽譜情報の記述を目的とした計算機用言語 (楽譜指向言語) を中心に説明する. 楽譜情報は音符 (と休符) の情報だけではないが, 音符の表記が最も基本となる. 今までに, 報告されている主な楽譜指向言語について, その音符情報の表記法を表 2.1 に示す. 音符には音高と音価が含まれる. 音高の表記法には, 大きく分類してオクターブと音名の組合せによる方法 (IML, Plaine and Easie, MUSTRAN 等) と, 五線上の位置による方法 (DARMS, COMSCORE) があり, 前者が多い. オクターブの表し方としては, 基準となるオクターブからの隔たりを記号 (例えば上を “” あるいは “+”, 下を “,” あるいは “-” で隔たりの数だけ記す) で表す方法と, オクターブを示す数値で表す方法が多い. 音名は英米式による通常の記法と同様に C, D, E, F, G, A, B で表す. 音価は数字 (IML, Plaine and Easie, MUSTRAN 等), 英字 (DARMS, Pla), あるいは特殊文字 (COMSCORE, music) で表記されている. 音価の数字による表し方には, 音価の逆数を表す整数値による方法が最も多いが, Plaine and Easie や MUSICA のように特定の数字により表しているものもあり, これは英字で表す方法と基本的に同じである. 音価を特殊文字で表す方法は, 音価が 2^n (n は整数) 拍であることを利用して, 基準となる音価の 2 倍と 1/2 を表す記号 (COMSCORE では “-” と “/”, music では “>” と “<”) により表す. 音高と音価は同時に表記する方法がほとんであるが, SCORE/MSS や Pla のように別々に表記するものもある.

次に, 音符以外の記号, たとえば音符の上下に書かれている各種記号や, 複数の音符にまたがって書かれている記号の扱いが問題となる. 音符に伴って表記される記号については, その書かれている位置で順序づける方法や, 記号の意味で順序づける方法がある. スラーのように複数の音符に

表 2.1: 音符の表記法例

言語名	開発者	音符の表記法 音高　音価 オクターブ　音名	例 	
IML (FASTCODE)	M. Jones (Princeton 大学)	UL C-B　逆数	L4C 4C	オクターブ変化を 上(U), 下(L)で表記
DARMS	S. Bauer-Mengelberg 他 (Columbia 大学他)	五線上　英字 の位置	19Q 6Q	音価: 倍全音符から B, W, H, Q, E, S, T, X, Y, Z
Plaine and Easie (ALMA)	Brook, Gould 他 (New York 市立大)	', C-B　数字	4'C 4"C	音価: 倍全音符から 0, 1, 2, 4, 8, 6, 3, 5, 7
MUSTRAN	J. Wenker (Indiana 大学)	+− C-B　逆数	4C, 4C+	音高, 音価共 連続値表記可能
SCORE/MSS	L. Smith (Stanford 大学)	数字 C-B　逆数	C4/C5, 4/4	パラメータごとに表記
MUSICA	Debiasi & de Poli (Padua 大学)	', C-B　数字	4'C 4"C	音価: 倍全音符から 5, 1, 2, 4, 8, 7, 3, 6, 9

かかる記号は, 記号の開始と終了で記述することが多いが, 音楽的には好ましくない.

ところで, 通常の計算機言語同様, 楽譜指向言語も手続的 (Procedural) 言語と宣言的 (Declarative) 言語に大別できる [Gou86]. DARMS に代表される初期の言語は手続的, すなわち記号を逐次記述してゆくものであり, 計算機への導入は比較的容易である. しかし, 記述の途中だけを見て, 記述の音楽的意味を理解するのは困難で, 意味を理解するにはそれより前の記述を遡って読む必要がある. これに対し宣言的言語では, 音楽を階層構造化し, 各階層は意味的にまとまりがある音楽オブジェクトの集まりで記述する. そのため, 記述全体の意味を理解するのにたどる文脈的情報の量は少なくてすむ. 音楽はそれ自体さまざまな構造を持っているため, 階層構造で記述することは自然に思える.

Ford 財団の援助を受けた Ford-Columbia Input Language (後の DARMS: Digital Alternate Representation of Musical Score) を初めとして, MUSTRAN (MUsic TRANslator), Plaine and Easie Code (改良版が ALMA: Alphameric Language for Musical Analysis), IML (Intermediary Musical Language) (後の FASTCODE) など, 1960 ~ 1970 年代にかけていくつかの音楽記述言語が開発されたが, これらはいずれも手続的言語である. これらに共通の特徴として,

- 楽譜を英数記号で表現する
- 楽譜上の記号の意味と比較的関連づけやすい英数記号で省略して表す
- 人間による入力を前提としており, その作業の負担を軽減するため省略表記を許す

の 3 点をあげることができる. またこれらの言語の目的は主として音楽分析であり, 印刷を目的とした DARMS も音楽分析用として用いられている. それぞれの言語は, 表面的には各楽譜記号に割当てられているコード (文字) が違うだけであるが, 設計思想にはそれぞれ特徴がある. DARMS は, 楽譜を単に記号が配置された図形として理解できれば記号化して入力できるように設計されている. 即ち, 入力者に音楽的な理解を要求しない. MUSTRAN は, 通常の五線譜では表現できない記号, 例えば半音より狭い音程の変化記号や, 音高や音価を連続量で表記することができる. これは, 民族音楽のコード化を考慮しているためである. ALMA は和音を音名, 音程のいずれでも表記可能で, さらに表記の順序も可能な限り自由にするなど, 柔軟な記述性に特徴がある. MUSICA は, 強弱記号, 装飾記号, テンポなど楽譜記号のほとんどのコード化が可能である (図 2.1). また, コメントを任意の場所に付けることができ, この内容に特別の意味付けを行うこ

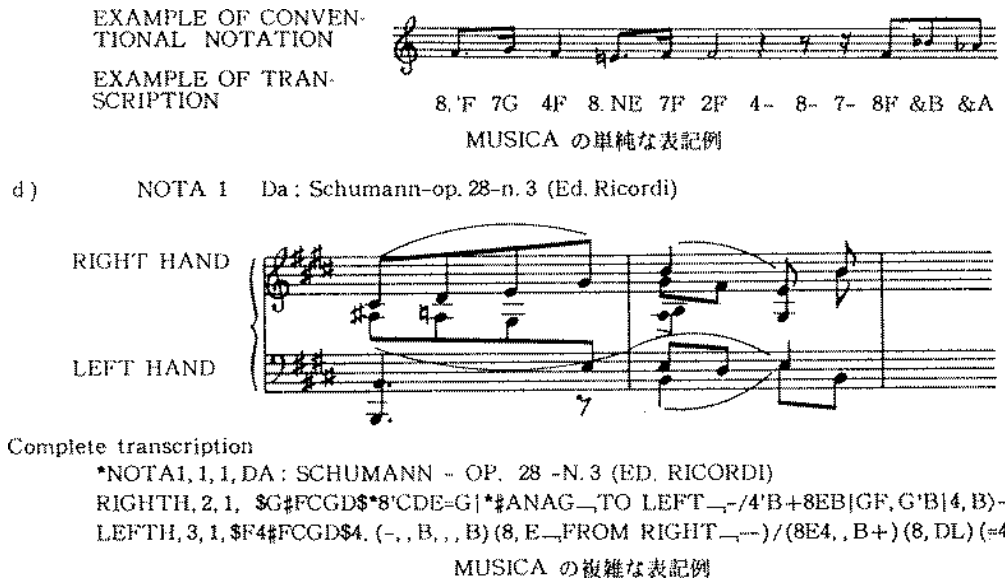


図 2.1: MUSICA による表記例

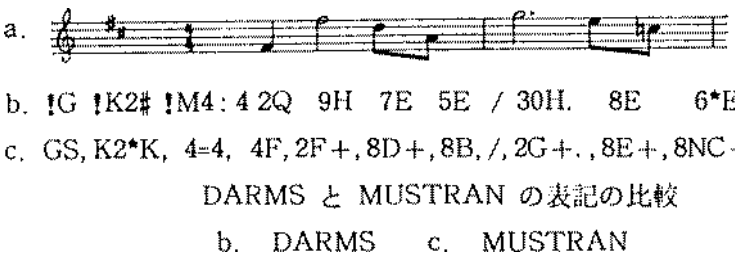


図 2.2: DARMS と MUSTRAN の表記の比較

とにより, 利用者自身での言語の拡張ができるようになっている. このように若干の差異は見られるが, これらの記述言語では, 表記法が非常に類似しているので (図 2.2) ここでは DARMS を例にその表記法について述べる.

DARMS は, 当初, 計算機による楽譜印刷を目的として開発された言語である. 図 2.3 は DARMS による楽譜の表記例である.

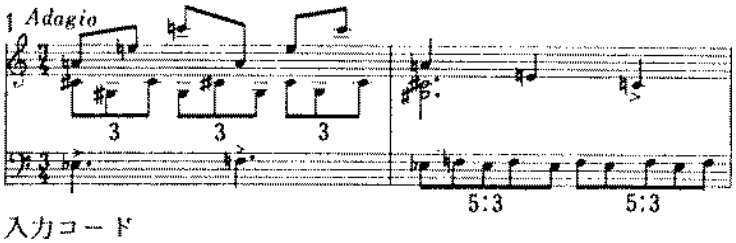
DARMS では,

1. 音高, 音価だけでなく, 楽譜中の全ての記号が表記できる.
2. 楽譜中の記号自体のコード化であり, 記号以外の情報, 例えば楽譜を解釈した結果などはコード化しない.
3. 単に記号の形を識別できさえすれば, 音楽の知識を全くもたない者でもコード化して入力が可能である.
4. 入力者の入力誤りを防ぐよう, 省略記法の導入や覚え易い記号を用いる等に配慮している.

などの特徴があるが, 他の言語と最も異なるのは 3 の点である.

DARMS では, 音高を space code という五線上の位置を示す番号で表し, 音価を特定の英字 1 字で示す. また, 印刷用のため, 符尾の向きは英字の U (上向き) と D (下向き) で表す. これらはすべて, 直前の音符と同じ場合には省略できる. 音符以外の記号, 例えば変化記号は# (シャープ),

楽譜例



入力コード

```
!I1!G, !F!M3:4, 18@\40\Adagio$!&!U4*E(8*)13*(4)
8(13) / 4*Q1*-1*) / &!D!R3-1#E (-4#, -14@\66\3$-1)
-4(-1#, -14@3$-4)-1(-4, -14@3$-1)$R / -1#H., -4#.
/$&$D!-50
!M3:4 6-Q.)7*Q./!R5 6-E(7*6,-4@5:3$ 7 6)7(6 7,-4@5:
3$ 6 7)/
```

図 2.3: DARMS による表記例

- (フラット), * (ナチュラル) のように, 音部記号は !G (ト音記号), !F (ヘ音記号) のように, 調号は !K に続けて変化記号とその数を, 拍子記号は !M に続けて分子, 分母の値を, 文字列は @ と $ の間に表記する. 連桁は (と) で囲んで表し, スラーはその開始と終了を文字 L に区別のための数字を付けて付記する. これらの記号を逐次表記するが, ピアノ曲のように複数の五線からなる楽譜や, 声楽曲での旋律と歌詞のように同時進行する情報の記述順序は, 五線毎でも小節毎でもよい.

これらの表記法は, 音楽のもつ階層的な構造を考慮しておらず, 単なる記号の羅列として表記するものである. また, 記号自体に与えられたコードについても, 理解が容易でないものが多い [Tal87].

宣言的言語の例としては, 米国数学会で正式に採用された TEX の利用を前提とする Gourlay の言語 [Gou86], TTS[Bal86], ANSI の標準化案 [Tal87] などがあり, いずれも近年報告されたもので, 音楽の階層的構造を考慮している. 以下, TTS を例に宣言的言語について述べる (図 2.4).

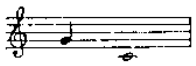
TTS (Twelve Tone String) は, 「いかなる目的にも使用できる言語は現実的でないし, また必要でもないだろう. 必要なのは異なる音楽処理の間でのデータ交換であり, 柔軟で音楽的にも自然な抽象化された枠組みを用いて, 音楽活動に共通な部分を標準化することである」という考えに基づいて設計されている. TTS 言語の基本単位は elementary TTS (p, d) で, これは音高 p と音価 d を表し, 楽譜上では 1 つの音符に対応する. p は primitive-element と呼ばれ, (オクターブ, (音名, 変化記号)) のように構造化されている. p には演算が定義されており, 音程を計算することができる. この elementary TTS あるいは (α||qβ) の形式をした構造 (ただし α, β は TTS, q は α の開始からの時間) もすべて TTS である. 継時進行を - 記号で, 同時性を | 記号で表すことにすると, q が α 自体の持続時間と一致するときは (α-β), q=0 のときは ($\alpha|\beta$) と記述できる. この継時性と同時性を示す 2 種類の基本的な記述を組み合わせ, 音楽を構造化して記述する. また, スラーやクレッシェンドのような記号も (α|<|) のように記述することにより, 音楽的に自然な表現ができる. このように TTS は音楽に含まれている旋律構造や和音構造を自然に記述することができ, 音楽分析用としては便利である. 一方, 印刷楽譜では楽譜記号の位置自体が芸術的意味をもっているが, TTS や Gourlay の言語では, 自動読み取りにより得られるような記号の正確な位置情報は表記することができない. ANSI の標準化案は, このような既存の表記法のもつ問題点を解決するものであり, 現在検討を重ねている.

TTS のような構造化された言語では, 旋律構造や和音構造のような音楽の構造に沿って記述

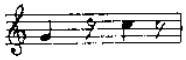
1. elementary TTS による音符と休符の表記

$p_1=\left((0,(5,0)),\frac{1}{4}\right)$, $p_2=\left((1,(1,0)),\frac{1}{4}\right)$, $p_3=((0,(1,0))1)$, $p_4=\left(\mathrm{REST},\frac{1}{8}\right)$

2. 簡単な TTS

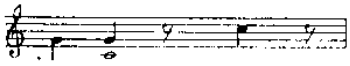


$\alpha=(p_1-p_3)$



$\beta=(p_1-p_4-p_2-p_4)$

3. 曲



$r=\left(\alpha \parallel_1^4 \beta\right)=(p_1-(p_1|p_3)-p_4-p_2-p_4)$

$(p_1 \mid p_2)$ denotes an *Harmonic interval*,

(p_1-p_2) denotes a *Melodic interval*,

$\left(\prod_{i=1}^{n} p_i\right)$, $(n\geq 3)$ denotes a *Chord*,

$\left(\frac{n}{i=1}P_i\right)$, $(n\geq 3)$ denotes a *Melody*.

図 2.4: TTS による表記例

できるため, 音楽的素養のある人には理解しやすく, コード化もしやすい言語と言える. 逆に, 音楽的知識のない者にとっては, かえって理解が困難になりかねない. 一方, あるがままの楽譜を音楽的な意味を導入せずに記述しようとする DARMS のような方法は, 一般的な使用を考えた場合には有効な面もあるが, 音楽の持つ構造が失われてしまう. 結局は DARMS も, 楽譜に現れる記号の種類が非常に多いため, 音楽的知識がないと全記号のコードを理解することが困難であったことも考えあわせると構造化した宣言的言語による表記法の方が音楽記述には向いていると思われる.

2.2 音楽情報の表現スキーム (平田)

計算機科学の分野では様々なプログラミング・パラダイムが提案され, それらを実現する様々なプログラミング言語や環境が研究開発されてきた. 音楽のような複雑な構造を持った情報をそれらプログラミング言語や環境でどのように表現すればよいか, またどのようなパラダイムが音楽記述に適しているかをサーベイする. 本章では Computer Music Journal の Machine Tongues シリーズを中心にサーベイを進める [Abb78a] [Abb78b] [Abb78c] [Roa79a] [Abb79] [Roa79b] [Kor80] [Kra80] [Lie82] [Lev84] [Pop89] [Dol89] [Fic91] [Hol92].

本節では取り上げなかったが, 他にも興味深い表現スキームとしてオートマトンを用いる手法 [Che92], 関数的集合的な手法 [Hir89], 分散協調モデル [Row92] などがあるが, 紙面の制約があるので触れない.

2.2.1 Lisp

Lisp は, 再帰関数の理論, ラムダ計算, ポーランド記法, リスト記法等に基づく大変柔軟な言語である. Lisp の枠組に沿って音楽情報の表現や処理を行うという試みを, 文献 [Kor80] に従って解説する. 最近は Lisp [Sch91] [Dan89] [Rah90] に限らず Scheme [Boy92] [NF91] や CLOS [Tau91] に基づくシステムも多く作られている.

データ構造

リスト構造を効果的に使うために, 我々が表現したい情報をリストを使ってどう具体化するかの手法を確立しなくてはならない. まず, 簡単な音楽作品をイベントのリストとして表現してみよう. 各イベントはリストで表現され, その最初の要素は音価 (duration) であり, 残りはその音価で演奏されたピッチ (pitch) である. 各ピッチはスケール度数 (C, D, E, F, G, A, B) と臨時記号 (シャープ, フラット, ナチュラル等) とオクターブから成る. 取り敢えず中央の C のオクターブをゼロとしよう. 二分音符の C マイナ和音はこのように表わされる:

```
(2 (C natural 0) (E flat 0) (G natural 0) (C natural 1))
```

この和音は中央の C とそれより上の合わせて 4 音を含む. この記法で, 良く知られたショパンのプレリュードのト音記号部の最初の 2 小節を表現してみよう:

```
((4 (E natural 0))
 ((8 dotted) (C sharp 1))
 (16 (D natural 1))
 (4 (E natural 0) (G sharp 0) (B natural 0))
 (4 (E natural 0) (G sharp 0) (B natural 0))
 (2 (E natural 0) (G sharp 0) (B natural 0)))
```

この記法が文字列として書くより優れている点は, リスト構造の特徴を活かして音楽情報の構造が表現されている点である. リスト構造で表現された音楽を操作する大変簡単な Lisp 関数は沢山考えられる:

1. 幾つかのメロディの断片を次々と一緒につなげる
2. 二つの部分から, AABA 形式を作る
3. 音列を逆転させる
4. 移調
5. 二つのメロディを対位法的に一緒にする
6. メロディのリズム間隔を変える

上の関数の中で, 例えば 1 は append として, 3 は reverse として殆どの Lisp システムで定義されている. 2 の機能を実現する 2 引数 Lisp 関数は次のように定義できよう:

```
(define Make-AABA (tuneA tuneB)
  (append tuneA tuneA tuneB tuneA))
```

この関数の本体は, 単に第 1 引数と第 2 引数のコピーをうまくつなげたものである. 上の最後の 3 つの関数も大変簡単に書けるが, 一応, さらに幾つかの (単純な) Lisp の概念を習得する必要があろう.

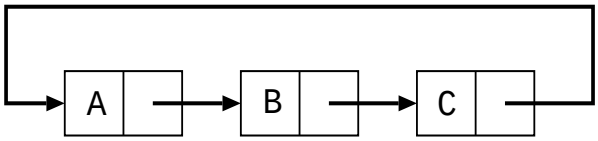


図 2.5: 無限リスト構造

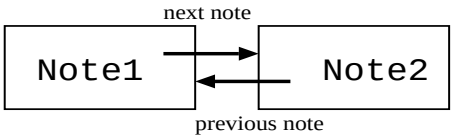


図 2.6: ポインタでつながれた 2 音

上記の例で用いられたデータ構造は完全に一般的という訳ではない. 暗黙裏に, 音楽的イベント同志はオーバラップしないことを仮定している. 即ちこの記法では, 次の音は前の音が止まってからでないと鳴ることができない. 筆者 (W. Kornfeld) は, クラシック音楽で記譜されるすべての音楽的イベントを記述できる, さらに一般的な記法を設計した. それは, ここで紹介した方法より多少複雑になってしまうが, 次のようなアイデアに基づいている.

リスト構造は, さらに複雑な構造を記述できる. リストはグラフ理論の木に大変良く似ており, さらに, 木が織り重なったような構造も表現できる. この織り重なり (interweaving) は, 音楽的イベント間の関係をより明確にするので重要である. 複雑な織り重なりのため, そのようなリスト構造は簡単にはプリントできない. 図 2.5 のようなリスト構造を実際作ることができる. これをプリントしたら次のような出力が得られるであろう.

```
(A B C A B C A B C .....
```

この出力は途中で stop をかけない限りずっと止まらない. この種のリストは **無限リスト** と呼ばれ, 実際 Lisp インタプリタの内部で生成されたりしているが, コンソールからリストとして入力されたり, 出力されたりすることは (一般に) できない. 無限リストの利用方法は幾つか有り, 双方向リストはその 1 つである. 例えば, ここに連なった 2 音があるとして, 音がお互い同志を指しあっている状況を考える (図 2.6). 音楽では, 種々のパートが多くの織り重なりの関係を形成している. 陽なポインタは, プログラムが意味的に正しい方向に動いていることを助けるので, プログラマの仕事を軽減する. 例えば, 図 2.7(a) のような 8 分音符と 4 分音符から成るリズムの断片を考えてみよう. まず, これらの各音には, 前の音 (preceding notes) と後ろの音 (following notes), 同時刻に開始する音という関係がある. 図 2.7(b) はこれらの関係をリスト構造を表現したものである.

Lisp の S 式 (リスト構造の文字列表現) は, 普通の言語で数値が扱われるように, 関数引数として, 関数の戻り値として, また変数の値としても使える. すると, メロディを引数として楽譜を値として返すような関数を書くことができる. この意味において, 我々は Lisp プログラミングを音楽システム記述に大変適した方法だと考えている. また, 筆者のシステムは, 基本データ型としてのメロディを取り扱うプログラムであり, 移調などの操作を基本操作として持つようなプログラムであると考えている. オブジェクト指向言語は, この考えをさらに進めたものとして捉えることができよう.

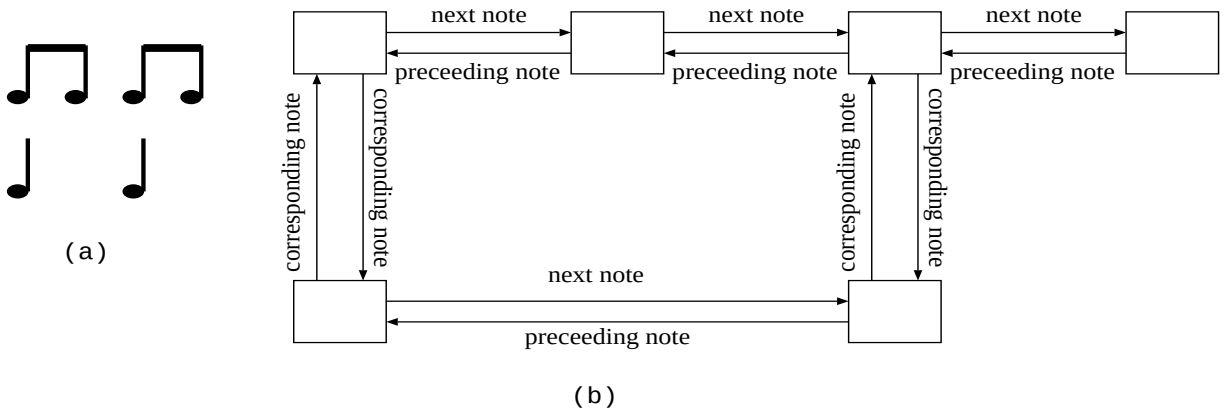


図 2.7: (a) 8 分音符のリズムと 4 分音符のリズム

図 2.8: (b) 対応するリスト構造

高階関数

Lisp では **全てのもの** がリスト構造で表現でき, 関数引数, 関数の返す値, 変数値になることができる. 関数自身にもこれは適用できる. `mapcar` は 2 つの引数をとり (関数とリスト), 返す値は引数の関数を引数のリストの各要素に適用した結果のリストである. 例えばメロディを移調する関数を書きたいとしよう. まず 1 音を移調する関数を書く. メロディは Lisp のリスト構造であるから, メロディ移調関数はこの 1 音移調関数と `mapcar` を用いて簡便に記述できる.

他に, Lisp は関数から関数への関数を定義することもできるし, `eval` (S 式の evaluator/interpreter) もあり, これらが, さらに柔軟で拡張性の高い Lisp プログラミングを可能にしている.

2.2.2 オブジェクト指向的表現

オブジェクト指向, オブジェクトベース言語によって音楽情報を記述する試みやシステムは数多く, IDP [Pop92a], SmOKe [Pop92b], HyperScore Toolkit [Pop87], Kyma [Sca89], MAX [Puc91], Animal [LdC91] 等々がある. 以下では Smalltalk と Act 1 を取り上げる.

Smalltalk オーケストラ

オーケストラは, 音楽を生成するシステムだと考えることができ, 種々の人々, 楽器, 楽譜などが統一の取れた方法で相互に連携し合っている. 計算機に対して, 音楽的なシステムを直接記述できないだろうか; 現実世界で見ているのと同じようにオブジェクトとインタフェースを定義できないだろうか? Smalltalk はこのような要求に合った言語, システムである [Kra80].

現実のオーケストラ: オーケストラの中央には指揮者が居る. 指揮者はオーケストラをリードし, オーケストラの演奏できるレパートリーを所有している. 指揮者はオーケストラに演奏開始・停止の指示を出す. 指揮者がある曲を演奏させたい時, 自分のレパートリーの中から望みの譜面を取り出して各パートをオーケストラの各団員に渡し, 演奏を指示し最後にその演奏を止める. オーケストラは演奏家の collection であり, 個々の演奏家は楽器とパート譜を持っている. 指揮者のレパートリーは楽譜の collection であり, 各楽譜は全ての楽器に対するパート譜を持っている. 指揮者の仕事の 1 つは, レパートリーの中から該当する楽譜を取り出し, 各パートを適切に演奏家に配ることである.

このオーケストラの記述では, 以下の 7 つの「物」に言及する:

指揮者 (Conductor): オーケストラをリードする人
オーケストラ (Orchestra): 演奏家の collection
演奏家 (Player): 音楽を生成する人
楽器 (Instrument): 各演奏家の持つ楽音生成器
パート譜 (Printed Part): 楽譜中の 1 つのパートだけを記した楽譜‘
レパートリー (Repertoire): 楽譜の collection
楽譜 (Score): 全楽曲の各パート譜の collection

上の記述を現実の Smalltalk 音楽システムに変換して行く. 必要な「物」を規定して, それからシステムを組み上げるために特定のインスタンスを選択するのである. この「物」を「オブジェクト」と呼び,「物」の規定を「クラス記述」と呼ぶ. 以下では, 各「物」について, 機能 (或いは期待される振る舞い) と内部表現に関してそのクラスを定義しよう.

オーケストラとレパートリーという 2 つの情報が指揮者を同定する特性である. Smalltalk では Conductor というクラスで指揮者を表わし, Conductor は orchestra と repertoire という 2 つのインスタンス変数を持つ. インスタンス変数はあるクラスのインスタンス毎に存在する. 指揮者がオーケストラに演奏せよ, 止まれなどと命令を出すことが, Smalltalk システムでは「メッセージ」によって定義される. メッセージはシステム内のあるオブジェクトから他のオブジェクトに送られる. クラス記述は, そのインスタンスが受信できるすべてのメッセージを規定している. 各メッセージに対応するのがメソッド記述 (アクション列) である. メソッドは, どのオブジェクトにメッセージを送るのか, どの変数値を変化させるのか, メソッド実行の結果としてどんな値を返すのか, 等を記述している. 指揮者は 2 つのメッセージを受け付ける: “曲を演奏する”, “演奏を止める”. 曲を演奏する時は, レパートリーに楽譜を取ってくるためのメッセージを出し, 楽譜に個々のパート譜を取ってくるためのメッセージを出し, 演奏家にパート譜を読んで演奏を開始するためのメッセージを出す.

ここに指揮者を Smalltalk で記述する時に必要な情報を全て書き出してみよう (インスタンス変数, メッセージ, メソッド).

Class: *Conductor*
Instance Variable: *myOrchestra* — the conductor's orchestra
Instance Variable: *myRepertoire* — the conductor's repertoire of scores
Message: *playAPiece: pieceName*

> **Method:** *playAPiece* というメッセージに演奏すべき曲を指定するための *pieceName* という引数をつけて送ると, 指揮者は *myRepertoire* にメッセージ *getAScore: pieceName* を送り, *myRepertoire* は対応する楽譜を返す. それから指揮者は取ってきた楽譜を *hereIsANewScore: score* というメッセージで *myOrchestra* に送る. 最後に指揮者はオーケストラに *playOn* を送信する.

Message: *stopPlaying*

> **Method:** メッセージ *stopPlaying* を送信すると, 指揮者は *myOrchestra* にメッセージ *stopPlaying* を送信する. 次に *myOrchestra* は全演奏家に演奏を止めるメッセージを送信する.

他の 6 つのクラスに関しても, 上と同様に, Smalltalk クラス記述のアウトラインを設計することができる. 原論文 [Kra80] では, 実際に他の 6 つのクラスに関してその振る舞いを書き下しているが, 本報告書ではその枠組だけ抜粋する.

Class: *Orchestra*
Instance Variable: *players*
Message: *hereIsANewScore: score*
Message: *playOn*
Message: *stopPlaying*

Class: *Player*
Instance Variable: *partName*
Instance Variable: *instrument*
Instance Variable: *printedPart*
Message: *getYourPartFrom: score*
Message: *playOn*
Message: *stopPlaying*
Message: *setInstrument: newInstrument*

Class: *Instrument*
Instance Variable: *envelope*
Message: *playNote: note*

Class: *PrintedPart*
Instance Variable: *partName*
Instance Variable: *notes*
Instance Variable: *position*
Message: *restart*
Message: *nextNote*
Message: *whatIsYourName*

Class: *Repertoire*
Instance Variable: *scores*
Message: *getAScore: pieceName*

Class: *Score*
Instance Variable: *pieceName*
Instance Variable: *pages*
Message: *whatIsYourName*
Message: *getPartFor: partName*

Smalltalk コード: ここで例としてクラス *Conductor* のメッセージ *playAPiece: pieceName* のコードを以下に示す.

```
playAPiece: pieceName | score part sheet each
  [ score ← myRepertoire getAScore: pieceName.
    myOrchestra do:
```

```
      [ :each.
        part ← each whatIsYourPartName.
        sheet ← score getPartFor: part.
        each hereIsYourMusic: sheet.
      ].
    myOrchestra playOn.
   ↑ pieceName
  ]
```

コード中, | の後の名前は, temporary variables の名前である. do ループは `do: [ ..code.. ]` で実現されている.

次の例としてクラス *Orchestra* のメッセージ *hereIsANewScore* のコードを以下に示す.

```
hereIsANewScore: score | eachPlayer
  [ self do:
    [:eachPlayer.
      eachPlayer getYourPartFrom: score]
  ].
```

システムの初期化: クラスのインスタンスを生成するために Smalltalk では, *new* メッセージをクラス *Conductor* に送れば良い. さらに生成された各オブジェクトを初期化するためには, 適当な引数で初期化のためのメソッドを起動すれば良い. つまり構文上は

```
ourConductor ← Conductor new.
ourConductor hereIsYourOrchestra: (Orchestra new).
```

のように書けば良い.

継承: まだ触れていない Smalltalk の重要な特徴がある. システム設計の段階で非常に重要な役割を果たす継承という特徴である. これは, あるクラスが他のクラスの特性や振る舞いを継承する機構である. 例えば, 室内楽オーケストラの使う楽器はフルオーケストラの部分集合であり, これはフルオーケストラにある制限が加わったバージョンであると考えても良い. また例えば, 打楽器奏者はオーケストラの他の演奏者とほとんど同じ振る舞いをするが, 通常 1 つ以上の楽器を演奏する. つまり打楽器奏者は普通の演奏者にある特性が付加された物と考えられる.

Smalltalk ではこの継承という機構を用いて, 差分プログラミングができる. 継承を行うクラスをサブクラス, 継承される (親) クラスをスーパークラスと呼ぶ. サブクラスではインスタンス変数やメソッド定義に関してそのスーパークラスとの差の部分のみ記述すれば良い. するとインスタンス変数やメソッド定義が新たに付加されたようなクラスが出来上がる. メソッド定義は上書きすることもできる. 以下に打楽器奏者のクラス記述例を示す.

Class: *Percussionist*
SubclassOf: *Player*
Instance Variable: *secondPartName*
Instance Variable: *secondInstrument*
Instance Variable: *secondPrinterdPart*
Message: *setSecondInstrument: newInstrument*
Message: *getYourPartFrom: score*
Message: *playOn*

ユーザインタフェース: Smalltalk のプログラミング環境の中心には, 情報は適切に表示しなければならないという考え方がある; 具体的には, ディスプレイ上の図, 表, アイコン, etc., 入出力デバイス, 特に音楽システムではオーディオ入出力などで, 慎重に設計されなければならない. Smalltalk の場合, システムの *Window* クラスに対して, *MusicWindow* というサブクラスを定義すれば比較的簡単にウィンドウプログラミングが可能になる.

いろいろな目的を持つ多くのユーザの理解や操作を助けるためには, ある情報が元は数値データの形で記録されていても, その情報をグラフィカルに多面的に (multiple views) 表示すべきであり, さらに画面上のイメージと実際のデータの間には因果関係 (causality) を持たせるべきである. Smalltalk でシステムを記述すると, このような要求は比較的容易に達成できる.

Act 1 を用いたオブジェクト指向的表現

ここでは, オブジェクト指向言語 Act 1 を幾つかの簡単な音楽システム記述に応用し, 音楽応用と人工知能の関連をオブジェクト指向という視点から考察する [Lie82].

何故オブジェクト指向プログラミングなのか: 音楽プログラミングと AI の間には少なくとも 1 つの共通点がある. AI と音楽の分野にとって, 効率というのはそれ程重要ではない. 問題の性質は, その解明が進むに連れて変化していくものであるし, 新しい制御構造やデータ構造が頻繁に現れてくる. AI や音楽プログラマは, 新しいアイデアが簡単に手早く検証できるようなプログラミング環境を必要としている. プログラミング環境は, プログラムに対する理解が深まるに連れてユーザが円滑に適合して行くように, データ構造, 制御構造に拡張性を備えるべきである. 実際, 革新的な音楽システムは常に "複雑さの壁" と戦ってきた. 作曲をしたり解析を行うプログラムは, すぐ複雑になってしまうので, プログラムの各部分の協調作業を適切に行うためには, プログラムのモジュラリティを上げる必要がある. 同様の要求は自然言語処理, ロボット, 画像処理, ゲーム, エキスパートシステムでも見られる. オブジェクト指向プログラミングは, システムをアクタと呼ばれるアクティブなオブジェクトの集まりとして組織することによって, プログラムを設計, 記述する手法である. 各アクタは他のアクタとは独立にアクションを行い, アクタ間の通信はメッセージ送受によってのみ実現される.

Act 1 は実験的なインタプリタ言語であり, 従来言語の全構成要素を, アクティブなオブジェクト (アクタ) とメッセージ送信で置き換えることができる. また Act 1 では, 多くのアクタが並列にメッセージを解釈実行できる.

オブジェクト指向プログラミングの利点: オブジェクト指向アプローチは, 知識を集中ではなく, プログラムの各部分に分散して表現することを可能とし, 従って知識とそれを操作する手続きを一体化することができる. Act 1 のようなアクタ言語では, アクタを操作できるのはメッセージ送信によってのみであり, これは multiple view の実現には本質的な役割を果たす. そしてオブジェクト指向におけるメッセージ送信の考え方は, 並列性に関して (細かいタイミングエラーを考えなくても良いという点で) 大変きれいな計算モデルをもたらす. Act 1 でイベントの順序に関する制約と言えば, アクタはお互い干渉することなくメッセージを送ることがある, という点だけである.

他のオブジェクト指向の特長と言えば, 共通な知識は共有できる, 一般的なオブジェクトからより特殊なオブジェクトを生成するためにメッセージの委譲ができる, インタフェースと実装が分離できる, 等があり知識プログラミングには欠かせない機構となっている.

アクタモデル: 基本アクタモデルは大変簡潔である. システムにはアクタという 1 種類のオブジェクトしか含まれない; 手続きもデータもすべて一様にアクタで表現される. アクタがメッセー

ジ (これもアクタである) を受信するとイベントが生じる. 各アクタは script (メッセージを受けとったらどのような動作をするか) と acquaintances (処理を行うために通信すべき他のアクタの名前) というデータ構造で表現される. あるクライアントのアクタ (client actor) の script がメッセージを受け付けられない時には, そのクライアントアクタは代理アクタ (proxy) にメッセージを委譲 (delegate) する.

1 つの音: 音楽システムを作る時まず問題になるのは, 1 つの音をどう表現するかである. 最も簡単な子供用のマイコン上の音楽システムでは, 1 つの音をピッチ (中央の C でゼロ, 半音毎に ±1 するような整数) と音価で表現することが多いだろう. しかしこれは 1 つの音を音楽とは独立と見なしている点で誤っている. 少なくとも調性音楽の範囲では 1 つの音はその楽曲のキー, 属するオクターブ, (キーに対する) ピッチ, 音価という情報を含んでいなければならない. さらに即興演奏を行うような用途では, その小節の和音やその 1 音が協和しているか否かの情報も含まなければならないだろうし, 音価に関してはスタッカートやレガート, さらに音色や音量を要求されるような場合もあるだろう.

このように 1 つの音は複合的なデータ構造をしているのだが, 次の問題はどうやってこれらの各性質をつなぎ合わせるかである; 配列, レコード, リスト等が使える. 実装上のジレンマとしては, 同じ情報でも, 使われ方によって適したデータ構造が異なる場合があるので, システムはある 1 つの元データ構造だけを保持し後は必要な度に変換を施すか, 必要なだけの種類のデータ構造を初めから用意しておく方法がある. しかし手続きの側から見ると, 計算機中にその情報がどのような形式で保持されているのかにはできれば注意を払いたくないであろう. これは multiple view の問題である. 従来の言語で multiple view を実現すると, 新しい性質を付加したりするのが大変困難になってしまい拡張性は悪い.

メッセージに対するアクタの反応: オブジェクトはメソッドあるいはハンドラと呼ばれる手続きによって, メッセージを受けとった時のアクタの振る舞いを記述している. 同じメッセージに対して同じ振る舞いをするアクタは, その実装が全く異なっていても交換可能である.

ここでは例として, 1 つの音に対して 2 種類のピッチを定義してみよう. プログラムは, Act 1 の細かい構文に立ち入らないために, 自然言語で記述する.

NOTE アクタ

acquaintances: PITCH, DURATION (全音に対する分数で表現), TIMBRE

これは PITCH, DURATION, TIMBRE というアクタを与えて NOTE アクタを生成することを意味している.

PITCH アクタ

SIMPLE-PITCH という名前の PITCH アクタ

インスタンス変数: OFFSET-FROM-MIDDLE-C

KEY-RELATIVE-PITCH という名前の PITCH アクタ

インスタンス変数: KEY, OCTAVE, OFFSET-FROM-KEY

次に, これらのアクタのメソッドを定義する.

もし自分が SIMPLE-PITCH で, 引数 LETTER [A–G], OCTAVE, ACCIDENTAL [♯, ♭, ♮] を持つ CREATE メッセージを受信したならば,

SIMPLE-PITCH アクタを生成し,
OFFSET-FROM-MIDDLE-C = OCTAVE × 12 + (middle C と LETTER の距離)
+ (もし ♯ ならば 1, ♭ なら -1)

もし自分が KEY-RELATIVE-PITCH で, 引数 LETTER [A–G], OCTAVE, ACCIDENTAL [♯, ♭, ♮], KEY を持つ CREATE メッセージを受信したならば,

KEY-RELATIVE-PITCH アクタを生成し,
OCTAVE = 引数 OCTAVE + (もし LETTER が C スケールで KEY より上ならば -1)
OFFSET-FROM-KEY = (middle C と LETTER の距離) + (もし ♯ ならば 1, ♭ なら -1)

次に, 例えば楽音合成ハードのために中央の C から半音でどれだけ離れているかという数値を得るためのメソッドを定義する.

もし自分が SIMPLE-PITCH で, ABSOLUTE-PITCH メッセージを受信したならば,
OFFSET-FROM-MIDDLE-C を返す

もし自分が KEY-RELATIVE-PITCH で, ABSOLUTE-PITCH メッセージを受信したならば,
KEY + OCTAVE × 12 + OFFSET-FROM-KEY を返す

このように同じメッセージに対して同じ振る舞いをするアクタが定義できた. 最後のメソッドの例を示す.

もし自分が NOTE で PLAY メッセージを受信したならば,
PITCH に ABSOLUTE-PITCH メッセージを送信し,
その返値を MUSIC-SYNTHESIS-HARDWARE に送信する

ここで注意すべきは NOTE の中の PITCH がどちらであってもこのメソッドは正しく動作するということである.

オブジェクトを用いたリスト表現: 多くのクライアントが同じ代理アクタを共有していることが有り得ます. 多くのアクタにとって, とても一般的な知識は代理アクタに, 特化された知識はクライアントに置くのが良いであろう. 委譲によって, 漸増的に知識や振る舞いを容易に拡張させることができ, この拡張は通常, 新しいアクタを付け加えるという形で実現される. 委譲は, Simula や Smalltalk のクラス, サブクラス, インスタンスという概念と置き換えることができ, オブジェクト間で知識を共有しかつ柔軟性を上げることができる. 他の言語では一般に, 知識共有のパターンはコンパイル時かオブジェクトが生成される時に静的に決まってしまう; これに対し, どの代理アクタにメッセージを送信するか (委譲するか) はクライアントが動的に決定する. Act 1 ではクラスとインスタンスはそれ程明確に区別されていない. すべてのアクタはクラスのように振る舞い, CREATE, EXTEND メッセージで自分と同じような振る舞いをするアクタを生成 (コピー) できる.

委譲に関する簡単な例を示す.

CHORD アクタ

インスタンス変数: LIST-OF-NOTES
PLAY メッセージを受信したら,
リストに連なる各 NOTE に同時に演奏せよというメッセージを送信する

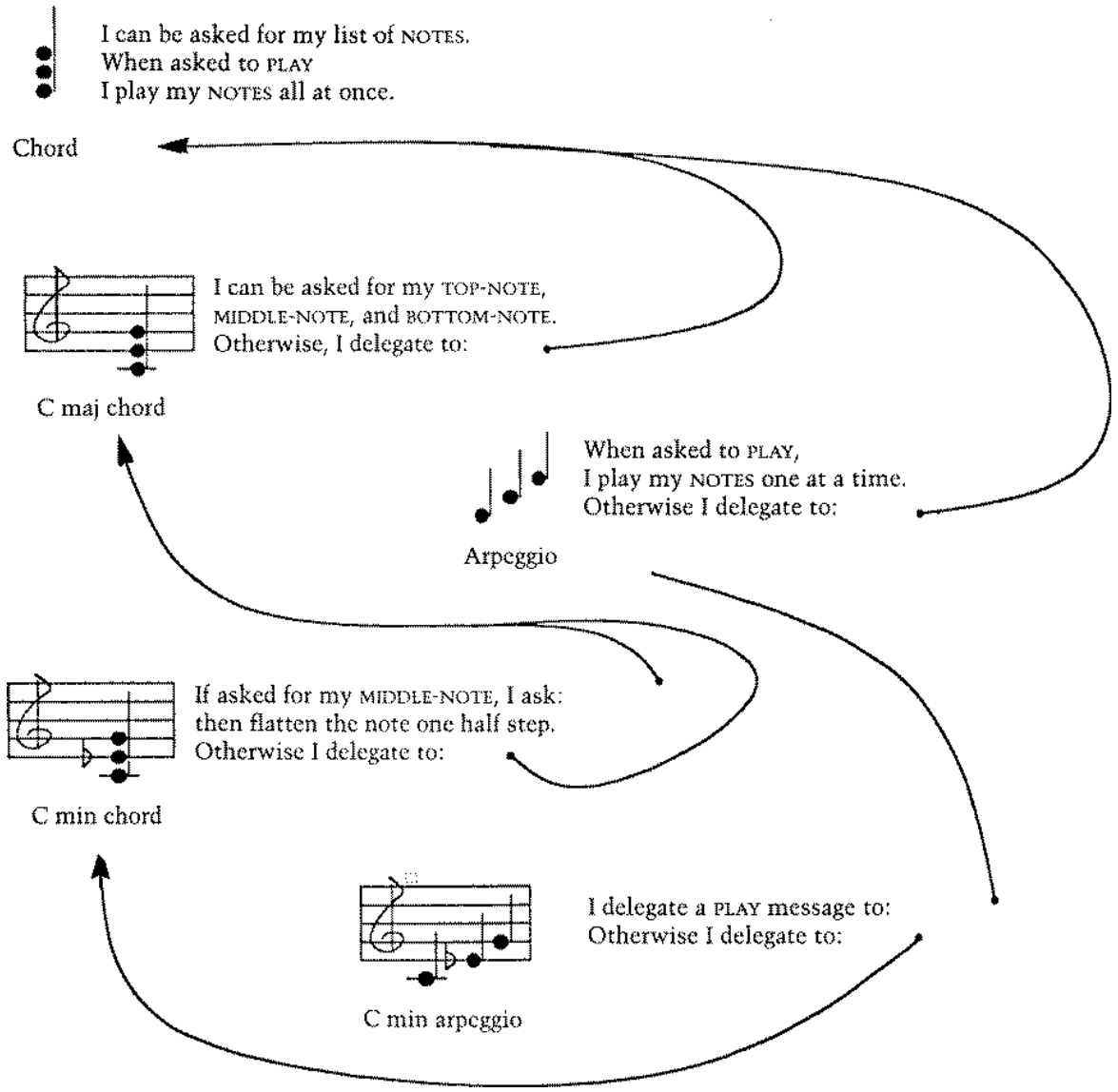


図 2.9: CHORD 間の委譲

ここで CHORD に EXTEND メッセージを送信してアルペジオアクタを生成する. 受信したメッセージの内, アルペジオアクタがインターセプトしなかった物が CHORD に委譲される. 各音のリストは共有できる点に注意.

ARPEGGIO アクタ

PLAY メッセージを受けとったら, 　　音のリストの最低音に PLAY を委譲する. 次に低い音に PLAY を委譲する, というのをすべての音について繰り返す.

次に MINOR-CHORD と ARPEGGIO というアクタを共有して, ARPEGIATED-MINOR-CHORD を生成する例を図 2.9に示す.

並列性:　知識がアクタシステム中に分散しているので, お互い干渉すること無く並行に実行することが可能である. また, 共有資源を操作するためにそこがボトルネックになってしまうような

こともない. アクタを各プロセッサにうまく分散させ, メッセージの送受信をプロセッサ間ネットワークを通して行えば良いだろう.

Act 1 が提供する並列プログラミングのための機構が 2 つある: future と serializer. future は並列プロセスによって計算される値を表現するアクタである. future は動的に生成され GC によって回収される. serializer は, 共有資源を包んでそれに対する操作を直列化する. この future と serializer は, 並列性をプログラマにとって transparent にしてしまう. つまりプログラマは, アクタが並列に動くことを意識せずにプログラミングできる.

キーボードで弾かれた音 (NOTE) をシンセサイザで演奏しつつ次々とディスプレイに表示することを考えよう.

> 引数に KEYBOARD-MELODY を与えて DISPLAYED-MELODY アクタを生成する:
> まず CONS アクタを生成し, それが KEYBOARD-MELODY の CAR に FIRST メッセージを送信する.
> 返値の NOTE に PLAY メッセージを送信する.
> 次に FUTURE アクタを生成し, それが NOTE に DISPLAY を送信して, 返値として DISPLAYED-NOTE アクタを得る.
> CONS アクタの CDR には, KEYBOARD-MELODY に CDR を送信して生成された DISPLAYED-MELODY を代入する

ディスプレイ上への表示とシンセサイザの演奏を future を使ってオーバラップさせる. future アクタは生成されたら直ぐリターンし, ディスプレイ上への表示と CDR 部の処理が並列に行われる. 他のアクタが future の値を要求した時 future の実行が終了していれば, その要求を受け付けることができるし, そうでなければ計算が終了するまで要求側を待たせる. 上の, アクタが値を要求するというイベントは, アクタシステムの中では正確に定義, 検出できる. この並列性と同期を常に正しく制御できるということで, 並列プログラムをより構造的に記述することが可能となる.

2.2.3 制約

制約プログラミングの枠組に立って音楽を記述する手法は, 実際はあまり提案されていない [TA91]. ここでは文献 [Lev84] に従って概観する.

汎用プログラミング言語を設計する幾多の試みの中から, 2 ~ 3 の制約プログラミング言語が生まれてきた. 制約に関するほとんどの仕事は, ある特定領域の問題を解く“エキスパート”をシステムに埋め込むことであった. 本稿では知識表現の一手法として制約を用いる. 制約は, ある意味で「問題解決アルゴリズムを実行する応用プログラムをその概念を真に理解している専門家から区別する物は何か?」という摑み所の無い質問の答えになっている. 制約システムでは, 通常の特定の問題を解く手続きを用意するのではなく, ドメインモデル (domain model, 部分や状況の間の関係, 依存グラフ) を構成している点が異なっている. 通常のプログラミング言語では, アルゴリズムをプログラムや手続きとして書くが, 制約言語では幾つかの異なる状況を組み込み制約やプリミティブな制約で関連付ける (図 2.10). ある問題領域の構造を記述するために, 組み込み制約で結合したネットワークを作る. この小さいネットワークの集まりをドメインモデルと呼ぶ. ドメインモデルはさらにことなる状況モデル (situation model) として結合されていく. ある状況モデルに質問を発すると, システムは内部の手続きの断片を統合して解を計算する.

TK!Solver

TK!Solver は Software Art 社の最近の製品である (論文出版は 1984 年). TK!Solver は基本的にはテキスト文字ベースのシステムであるが, グラフィック機能も持っておりプリミティブ制約を任意に選択しそれらを任意に結び付けることができる.

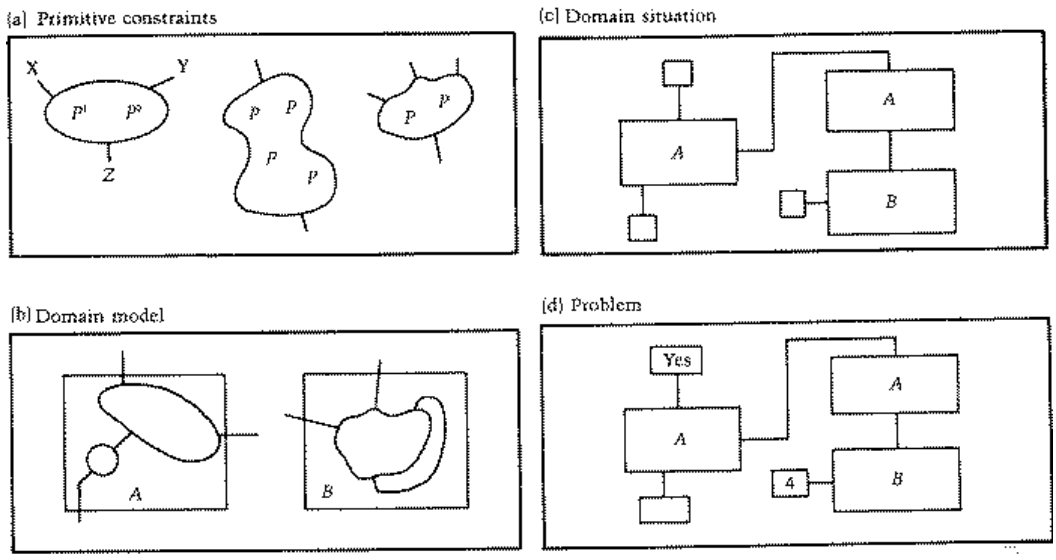


図 2.10: 制約のメカニズム

TK!Solver では $D = V \times T$ のような式がいろいろに解釈される. ここで明記したいのは, 代入のための = と制約の = は違うということである.

TK!Solver は汎用プログラミングシステムではない. SketchPad のように, 2 つの算術 (arithmetic), 三角法 (trigonometric) とその他定量的な関係を指向した組み込みアルゴリズムを持っている. つまりこれらは数式を扱えるわけであるが, 多くの興味深い定性的あるいは記号的な制約は解くことができない.

音楽のモデル

モデル駆動の問題解決は, 特に音楽 (作曲と分析) の分野では有効である. このことは, 著者 (D. Levitt) が計算機によるジャズ即興の研究を開始した時 [Lev81] にすでに明らかであった. 例えば, 即興を生成する場合は, 実際のメロディ, 和音, ケーデンスなどが決定される以前にメロディのピッチはその和音の 3 度になっていなければならない, というような制約が課せられるのである. さらに作曲をする場合などでは, 音楽理論とスタイルの制約をうまく組み合わせる必要が生じる. さらに作曲が進むと, ある 1 つの決定が次々と伝搬してメロディや和音が決定したり, 制約の状態が変化を受けたりする. このような領域を表現するために我々の言語は, 任意に記号や記号間の制約を定義することができないといけない.

ここで簡単な協和, 不協和の例を示す. ここでまず和音を集合的に定義する.

```
((AllPitches = {C Db D Eb E F Gb G Ab A Bb B}
  AllQualities = {MAJOR MINOR}
  Pitch = Member(AllPitches
                 Quality = Member{AllQualities}
                 Chord = [Root<a Pitch>, Quality<a Quality>]
```

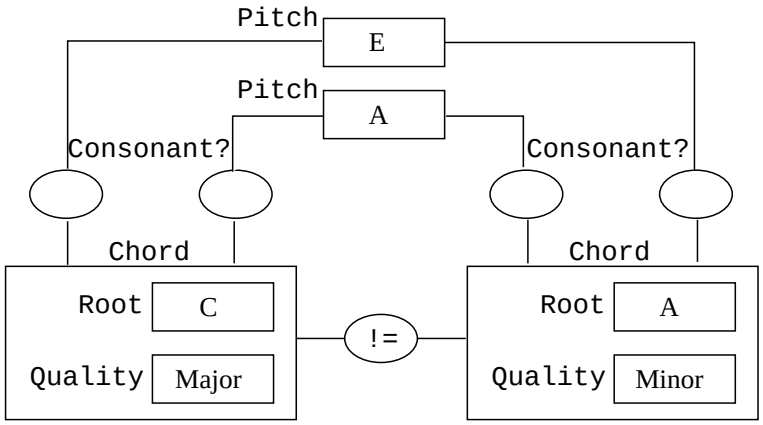


図 2.11: 状況モデル

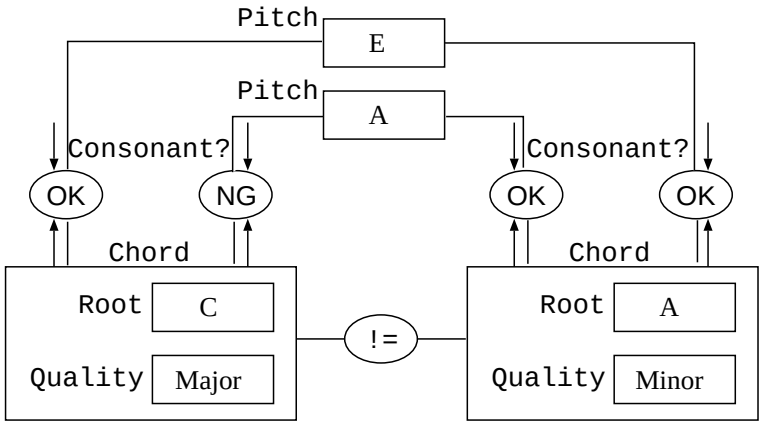


図 2.12: 状況モデルへの問い合わせ (a)

```
MAJOR = {Unison Major-3rd Perfect-5th}
MINOR = {Unison Minor-3rd Perfect-5th}
Consonance = [Chord<a chord>, Pitch<a pitch>,
              Consonant?<a Boolean>])
))
```

根音 C, D$\cdots$ は 12 のモジュロで表わす. このようにして 24 個の和音からなるドメインが定義される. 次に和音と音との間の関係を記述したネットワーク, つまり状況モデルの例を図 2.11 に示す. 図 2.11 では, 2 つのピッチ (E と A) が consonant? という制約 (楕円で示してある) で和音 (Chord) と関係付けられている. 2 つの和音間は “等しくない (!=)” で関係付けられており, これは 2 つの和音が直列か並列に並べられていることを意味している.

さらに図 2.12, 2.13, 2.14, 2.15 と異なる問いを発した時の状況モデルの動作を順次示している. 図 2.12 では和声を分析する簡単な問題を示す. この制約ネットワークは, E は C Major 和音と協和するか? A とは? などの質問を構成している. Constraint? というノードでは, システムの提供するある述語で, (和音の構成音に関して) メンバーシップ試験が行われ真偽値が計算される. 図 2.13 では同じネットワークに対して, 別の制約問題を解かせてみよう. ユーザがすべての音は協和しているという仮定を課す. もし協和する和音に選択肢があればユーザはそれらの中から選

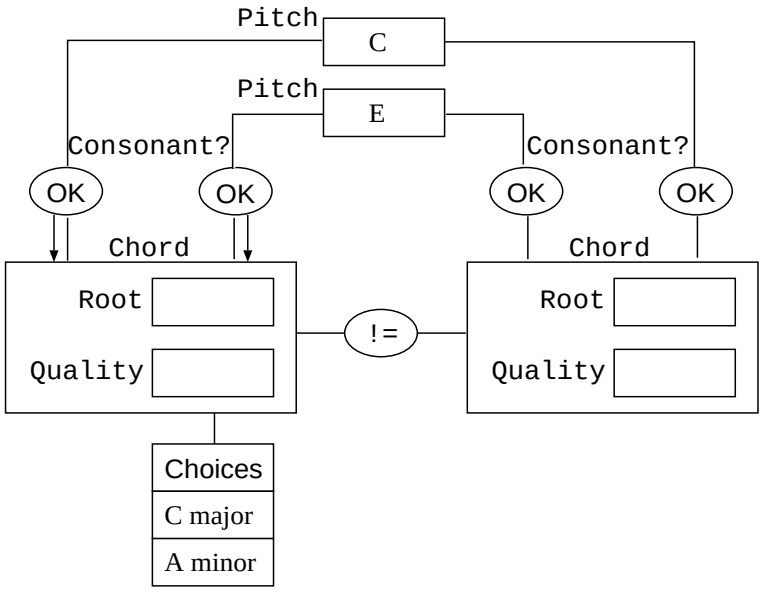


図 2.13: 状況モデルへの問い合わせ (b)

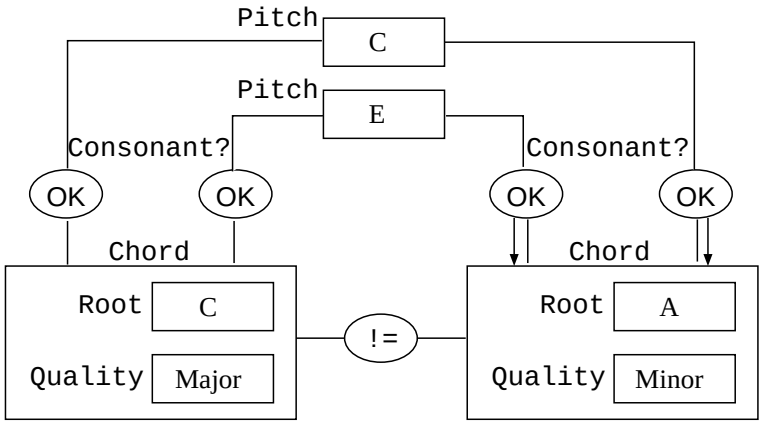


図 2.14: 状況モデルへの問い合わせ (c)

択できる. 図 2.13 で C Major を選択すると, 図 2.14 のように制約が伝搬して右側の和音が A Minor と決まる. ユーザが, 今度は逆に「C Major と A Minor 両方に協和する音は何か?」という問いを出すと 2 つの候補が提示される (図 2.15). 普通ユーザは, システムが答や代入をどのように計算しているかを気にする必要はない. ユーザが行うのは, ドメイン上の概念 (和声理論) をよく理解されている概念 (数の計算や集合演算) にマッピングすることだけであり, そうすれば後はシステムが適切な手続きを適応してくれる. 例えば図 2.14 では, “もし A が要素が 1 つしかない Set オブジェクトの要素なら, A が唯一の要素である” という手続きが用いられているかもしれないが, それは言語のインタプリタやコンパイラの設計において決定される事柄である.

依存関係とデバッガ

AI 的なプログラムとそうでないプログラムを区別する点の 1 つは説明能力である. ほとんどの制約言語システムは, 出てきた結論を計算に用いた制約などを使って正当化することができ

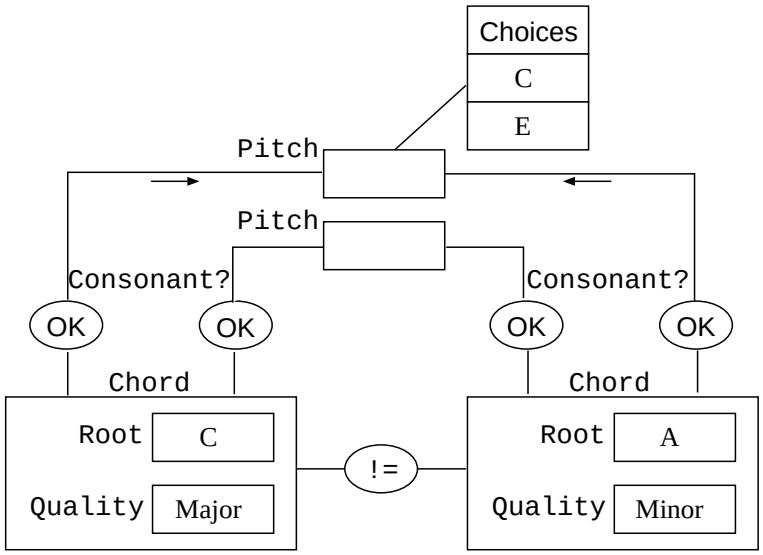


図 2.15: 状況モデルへの問い合わせ (d)

る．さらにユーザの指示で, ある結論がどこから導かれてきているのかを辿ることができる．これが状況モデルが依存ネットワークと呼ばれる由縁である．例えば図 2.14 で A マイナ和音を正当化せよと命令したら, 図 2.16 のような応答が得られる．システムはユーザからの “WHY?” という質問に答えていく．図 2.16 の各レベルは, 次々と発せられる why? という問いに対してネットワークの関係する部分を参照している．このような言葉による説明は, 問題記述やモデルのデバッグには非常に有効である．図 2.17 は一貫性の無い (inconsistent) 過制約 (overconstrained) な状態をシステムに与えた場合の挙動である．システムは, ユーザの課した制約を 1 つずつ吟味していく．図 2.17 (b) ではシステムが過制約を検出し, 左側の和音を削除 (retract) する．図 2.17 (c) ではシステムが制約を計算して, 他の候補を幾つかユーザに提示する．

ある結論を削除した時, ネットワークを通じて削除が伝搬して行く (dependency-directed reasoning, 依存に基づく理由付け) ような機能は, 知的振る舞いをするシステムにとっては重要な役割を果たす．

反復による制約充足アルゴリズム

制約言語システムには, 複数の制約充足アルゴリズムが組み込まれ, 自動的に協調動作するようになっている．ThingLab や TK!Solver は汎用性の高い反復による制約充足アルゴリズムを備えている．アルゴリズムの概要はこのようになっている:

1. 解の推測を 1 つ取る
2. その推測から導かれる結論を伝搬させる
3. 結果が矛盾していないか, 制約を満たしているか検査し, もし OK ならその推測を保持する．もし駄目なら推測を削除して別の推測を立てて 2. へ行く

しかし反復法も万能ではないので, 効率的に適用できるというもっと大局的な判断のもとで使うべきである．

また問題を効率良くためには, 冗長なビュー (redundant view) を持つことも必要である．例えば算術の領域であれば, 変数を消去した連立方程式を計算するとか, 和は差の逆演算であるといった (冗長なビューを計算する) 知識を用いた方がはるかに効率的に解が求まる．

```
WHY?
The chord is A minor because
  there are 24 chords, and
  23 chords are ruled out, and
  A minor is not ruled out.
  WHY?
    Because
    the chord must be a member of AllChords, and
    Consonant? in Consonance-3 must have a value of YES, and
    Consonant? in Consonance-4 must have a value of YES, and
    the chord must be NOT-EQUAL to C Major,
    but,
    AllChords is a Set of 24 chords, and
    14 are ruled out because
      both
        the value of Consonant? in Consonance-3 would be NO
          and
        the value of Consonant? in Consonance-4 would be NO, and
      4 others are ruled out because
        the value of Consonant? in Consonance-3 would be NO, and
      4 others are ruled out because
        the value of Consonant? in Consonance-4 would be NO, and
      1 other is ruled out because it it not NOT-EQUAL to C major.
      A minor is not ruled out, so the chord is A minor.
```

図 2.16: A マイナ和音の正当化

制約と演奏

音楽理論の問題は, 制約言語にまた別の機能を導入する必要があった: それは, 音楽イベントの分析や構成のために, 新しい記述 (オブジェクトや依存関係) を導入していく機能である. 我々は協和性 (consonance) というオリジナルな制約を定義し, この関係を使って分析や作曲を行うことができた. このようにして, ある意味でアーティストのスタイルを制約というボキャブラリとして定義したのである.

音楽において, Levitt のジャズ即興プログラムはコード進行とメロディからソロを生成するために, 和声, メロディ, テーマの相互的な制約を充足させる. このシステムは 30 の音楽的言葉 (最も基本的な音楽理論用語や音楽的制約関係) を持つ. これだけでも現在の制約言語システムの能力を越えているように思える. 例えば相互制約 (mutual constraint) は, 構造的な探索のための高レベルの制御を与えることができるだろう.

2.2.4 ニューラルネットワーク

ニューラルネットワークの計算モデルには, プロセッサとメモリという二分法的考え方や記号操作という概念が無い. その代わり, まず本質的には理想化されたニューロンのアナログ的な

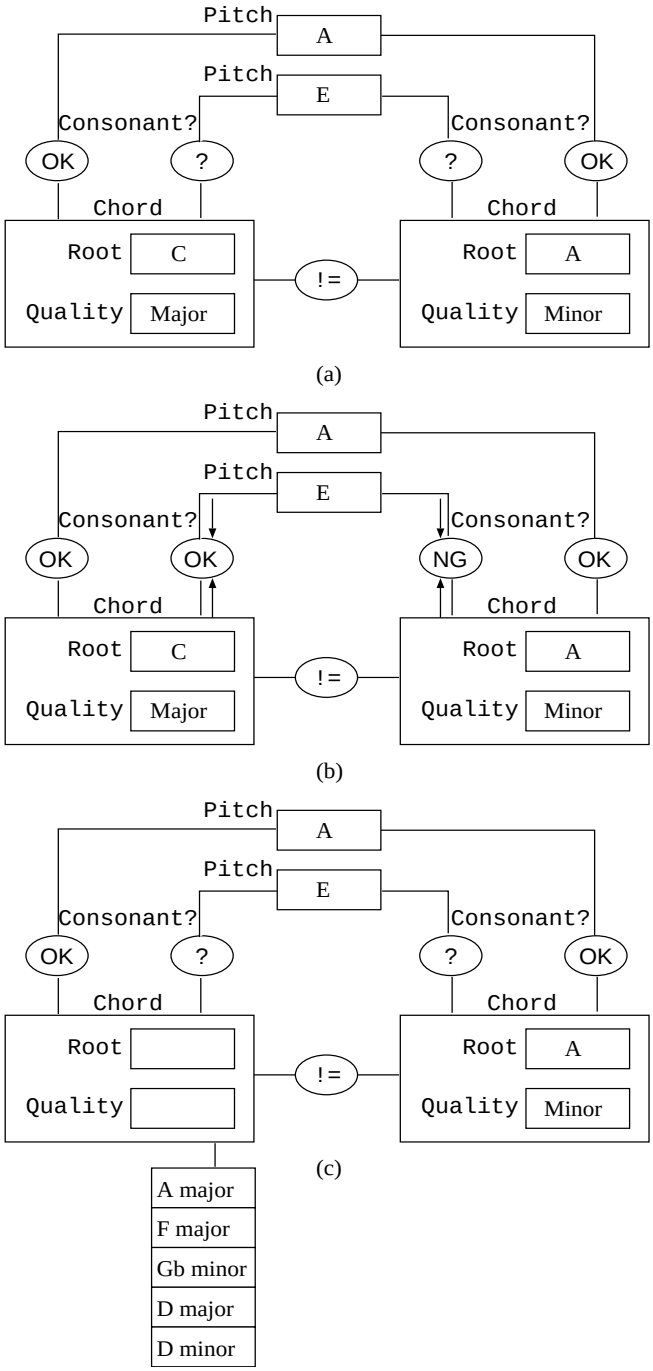


図 2.17: 矛盾を見つけた場合の動作

ネットワークがあり, その中ではニューロン間接続の関数として異なる興奮パターンが存在する. 処理とメモリはネットワーク中に一様に分散されており, 並列マシンに効率良く実装されるような計算モデルと言えよう. 最近では文献 [Tod92] のような研究もある. ここでは文献 [Dol89] に従って解説する. Computer Music Journal (MIT Press) Vol.13 No.3, 4 (1989) で特集を組んでいるので興味のある方は読んでみると良いだろう.

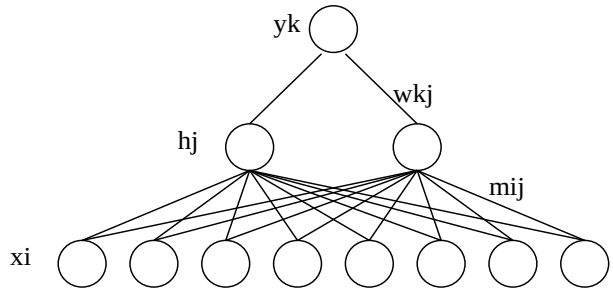


図 2.18: リズムの好ましさを判定する 3 層ニューラルネットワーク

例: リズムの評価器

簡単なリズムについて人間の良い悪いの判定をエミュレートするソフトウェアを作ることを考えよう．もちろん簡単なルールで好ましさを記述できるならば, リズムを良い悪いに分類する if-then 文の集まりでプログラムを書けるだろう．しかし往々にして, 何故それが好ましいのかということは分からないのに何が好ましいのかは分かる．このような時にニューラルネットワークという手法が有効である．

与えられる例題は 4/4 拍子, 8 分音符より速い音符は無いと仮定する．すると 1 小節内には 8 個の音 (attack or rest) があって, 1 個目だけには必ず attack が来て, 2 拍目 4 拍目は両方とも attack か rest であるとしよう．また, リスナの好ましさを決定しているルールは存在しているが, それはリスナには良く分からないとしよう．この問題はどのようにしてニューラルネットワークで定式化すれば良いのだろう?

様々な選択が有るが, ユニットを効率良く使いかつニューロン間の結合が解くべき問題と整合しているような定式化が望ましい．まず入力は 8 個の拍であるから, 8 個の入力ユニットを設けよう．1 個の出力ユニットは 1.0 (良いリズム) か -1.0 (悪いリズム) を出力する．現在, 中間層 (hidden layer) の最適な個数を決定する方法は見つかっていない．一般に, 中間層のユニット数が少ないと計算は軽くなるであろうし, 得られる結果もより一般的になるであろう．この例では 2 個と定めておく．結局, 図 2.18 のようなニューラルネットワークを得る．重みに関しては図に示してある 18 個に加えて, 中間層と出力ユニットの 3 個のニューロンは補助バイアスを得るためにもう 1 本ずつ多く入力を持っており, 全部で 21 個の重みが必要となる．そして, このニューラルネットワークをバックプロパゲーションアルゴリズムによって学習させれば良いのである．バックプロパゲーションについては紙面の都合もあり, ここでは深入りはしない．

一般化の問題

図 2.18 には 8 個の入力があるので, 単純には 256 通りの入力パターンが有り得る．しかし一般にニューラルネットワークには 256 通り全部の例が与えられる訳ではない．できれば, より少ない例からより少ない時間で同じ学習結果を得たい．即ち, 少ない例から得られた学習が, まだ示されていない未知のパターンに対しても正しくあって欲しい訳である．どうやってこの一般化の問題を解決すれば良いのであろう? 実際, これはニューラルネットワークの重要な研究テーマとなっている．

表 2.2 は, 12 個の良いパターン, 28 個の悪いパターンで学習した結果得られた重みである．行 i 列 j は, ユニット j からユニット i への重みである．例えばユニット O1 は H1 からの信号を重み -5.61 で受けとる．ネットワークには最初ランダムな重みを与えておき, 二乗誤差が 0.0002 以下になるまでバックプロパゲーションを繰り返した．これには 200 回の繰り返しを要した．さて

表 2.2: Feed-foward ネットワークの重みマトリックス

	BIAS	I1	I2	I3	I4	I5	I6	I7	I8	H1	H2	O1
H1	-0.24	-5.35	0.01	5.65	1.55	1.30	-0.15	-5.57	-0.32	0.00	0.00	0.00
H2	-0.54	-3.50	-0.77	-6.65	-0.23	0.20	0.63	6.05	-0.57	0.00	0.00	0.00
O1	-5.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-5.61	-5.65	0.00

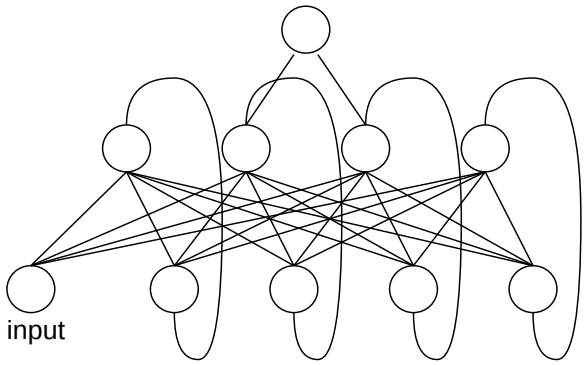


図 2.19: 繰返し構造を持つネットワーク

この学習がうまく行っているかどうかを検証するために 256 パターンの内, 条件を満たすすべてのパターンについて調べたが, すべて OK であった.

時間の問題

次に, ある楽曲の全体について, その中に好ましくないリズムパターンが存在するか否かを判定するネットワークを考えよう. もし上に挙げたようなネットワークで楽曲全体を一度に入力するためには, 膨大な数の入力ユニットと学習時間が必要となる. そこで一度には 2~3 小節だけ聴きながら好ましいか否かを判定するネットワークを考える. この何小節かの中には, 好ましくない最長のフレーズが収まらないとならない. しかし, 往々にして殆どの好ましくないフレーズは短く, その場合余っている入力ユニットは雑音源となってしまう.

もう少しうまい方法として, 入力のリズムを逐次的に与えてみよう. しかし普通の feed-forward のネットワークでは, その時点で与えられた入力に基づく出力しか得られない. どうやって進んでいく時間という概念を組み込むことができるのであろうか? そこである層から下の層に遅延 feed-back のできるネットワーク, 例えば, 時間 t における入力の一部は, 時間 t-1 における中間層か出力ユニットから供給されるようなネットワークを考えよう. 基本的にネットワークは feedforward で動作しているのであるが, 同時に以前の情報を再利用して内部状態を更新させるという風にも見なせる. この場合, 厳密に言うとバックプロパゲーション学習できなくなるが, その改良された学習方式が幾つか提案されており, 中でも Williams と Zipser が提案した real-time recurrent learning (RTRL) [WZ88] は今考えているようなニューラルネットワークには適している.

例えば 3 個以上の 8 分休符が連続するか 5 個以上の 8 分音符が連続すると好ましくないとしよう. この判定には図 2.19 のようなネットワークを用いる. 1 個の外部入力ユニット (attack は 1.0, rest は -1.0) と 1 個の出力ユニット (好ましくないと -1.0, それ以外は 1.0 を出力する) を持つ. 各時間ステップで中間層は 4 個の (1 ステップ分の) 遅延ユニットから出力を受けとる. 簡単のために出力ユニットは中間層のある 2 個のユニットから入力を得る. このネットワークに 96

表 2.3: 繰返しネットワークの重みマトリックス

	BIAS	I1	H1	H2	H3	H4
H1	-1.67	1.34	0.77	2.97	1.28	-0.38
H2	0.88	2.67	2.12	-1.72	0.58	-1.64
H3	1.28	-2.39	-1.65	-2.62	3.66	1.68
H4	3.71	3.53	0.57	1.50	-1.22	4.25
O1	-4.00	0.00	0.00	0.00	6.15	6.15

表 2.4: 4 つの入力リズムパターン

#1	1	-1	1	-1	1	-1	-1	-1
#2	1	-1	1	-1	1	1	1	-1
#3	1	1	-1	1	1	1	-1	-1
#4	1	1	1	1	1	-1	1	-1

個の入力例を与え RTRL を行った結果を表 2.3 に示す. 行 i 列 j は, ユニット j ユニット i への重みを示す. 例えばユニット O1 は H3 から 6.15 の重みで信号を受けとる. この重みの表を眺めて解の意味を探るのは困難である. ちょっと考え方を変えて dynamic system analysis の分野の技術を応用する. 出力ユニットが 1.0 を出力するのは,
$6.1 \times unit_3 + 6.1 \times unit_4 > 4.0$
の時であり, この式をもとに phase portrait を描くとより理解を深めることができる.

例: 計算機支援作曲

最後にニューラルネットワークで自動作曲 (リズム生成) を行ってみよう. 例えば Todd の研究では [Tod88], 学習フェーズで単に好ましい多くのリズム例をネットワークに提示し, 学習完了後にバリエーションを限りなく生成させるようにするという単純な方式がベースになっている. ここでは, 図 2.19 のネットワークに表 2.4 の 4 つの好ましいリズムパターン時系列を提示する (1 は attack, -1 は rest である). この 4 つのパターンについて, 与えられた入力拍の次の拍が出力されるように学習する. 学習が完了したら, 出力を 1 ステップ時間遅延させて入力に加えリズムパターンを生成させる. 出力ユニットの時系列パターンは, 学習のために提示された入力パターンに似ている筈である.

ではネットワークに 4 つのリズムパターンを学習させるにはどうすれば良いのだろうか? 簡単な方法としては入力ユニットを 1 個から 4 個に増やし, i 番目のパターンは i 番目の入力ユニットにのみ提示することが考えられる ($1 \leq i \leq 4$). 出力ユニットのパターンをどの入力ユニットに戻すか (動的に) 変えることで, リズムの混合やフェードイン / アウトができる. 表 2.5 は 4 つのリズムパターンを学習した後の重みを示している. 我々のネットワークを Todd のものと較べると, 中間層からフィードバックがかかっている点が大きく異なっている. Todd のネットワークにはバックプロパゲーションが適用できるが, 我々のネットワークでは駄目である. しかし時系列イベントを扱うような問題にはとても柔軟に対処できるという利点があろう.

表 2.5: 4 つのリズムパターンの学習結果

	BIAS	I1	I2	I3	I4	I5	H1	H2	H3	H4	O1
H1	-1.34	-0.34	1.00	-2.20	-1.12	2.07	3.57	-1.97	0.88	3.51	0.00
H2	1.66	-0.57	-0.47	1.91	-0.76	-l.24	-2.70	2.67	1.55	-0.98	0.00
H3	-1.70	-0.33	4.29	-2.39	-2.58	3.24	-1.44	-0.74	-2.14	3.45	0.00
H4	-1.02	-3.23	-3.24	3.54	2.98	5.27	3.29	-8.10	5.61	-4.30	0.00
O1	6.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.47	-14.33	0.00

将来の展望

本稿で扱ったようなリズム自動生成ネットワークをはじめとして, ニューラルネットワークはコンピュータミュージックに大きなインパクトを与えた. 以下 2 つばかり他の例を挙げよう.

音色を決定する基本となる楽音信号の特徴や次元を同定するためのニューラルネットワークという研究が進んでいる. 図 2.19 のネットワークに楽音信号を (時系列的に) 入力し, それぞれ異なる楽器に対応する複数の出力ユニットを持つように拡張すれば, 音色弁別器として機能するようにもできよう. ただし各出力ユニットから出た信号が “どのような音色” に対応しているか (エンベロープ, 音色の明るさ, etc.) は分からない.

waveshaping と呼ばれる楽音合成方式では, 単純な周期信号はある非線形関数の拡張としてみなせ, その非線形関数をうまく選択すると, そこから任意の周期関数を得ることができるという方式である. 図 2.18 のタイプのネットワークを, 1 入力ユニット, 1 出力ユニット, 多くの中間層, (1.0, -1.0 ではなく) 連続値を扱えるように変更すると, waveshaping のための (学習できる) 非線形関数として振る舞うことができる. さらに, 音量や音高も扱うための入力ユニットを付加すると, 周期的に変化する入力を与えた時, ネットワークは望ましい周期波形を出力するだけでなく, 音量や音高の変化に応答して波形を変化させることもできるであろう.

2.2.5 CSP – Communicating Sequential Processes

並行計算のモデルはいろいろ提案されてきたが, その中でも最も有名なものの 1 つが C.A.R Hoare によって 1978 年に提案された Communicating Sequential Processes (CSP) であろう. 簡潔なモデル, 記述の簡便さ, エレガントなプリミティブ等がその理由であろう. CSP 自体はプログラミング言語というよりは, 抽象的プロセスや並行動作を記述するための形式的計算モデルといった雰囲気が強い. ここでは CSP を音楽記述に応用した例を紹介する [Hol92].

定義

以下 CSP の概説を行う.

入出力コマンド　CSP では入出力はプリミティブであり, 1 対 1 通信である. 入力は `P1?x`, 出力は `P2!y` のように書く. ここで `P1`, `P2` はプロセス, `x` は変数, `y` は式である.

並列コマンド　2 つ以上のプロセスを並行に実行する場合は, `[P1 || P2 || .... || PN]` と書く. `Pn` は単純プロセスでも複合プロセスでも良い. 並行に実行されるプロセス間では共有変数が持てない.

ガードコマンド　条件実行はガードコマンド `G → S` で実現する.

alternative コマンド, 繰返しコマンド　非決定的な選択は alternative コマンドで以下のように記述する. [G1 → S1 □ G2 → S2 □ .... □ GN → SN]

繰り返しは, *[G1 → S1 □ G2 → S2 □ .... □ GN → SN] のように記述する. すべてのガード (G) が失敗するとこの繰返しを抜ける.

他にもまだ説明していない構文 (例えば代入 x := y;) がある. 以下, CSP による並行プロセスの記述を音楽的アクティビティの記述に利用した例を幾つか挙げるので, その中で適宜取り上げ解説する.

MIDI Transposer の例

MIDI データを読み, ピッチを 1 オクターブ上げ, 出力するという 3 つのプロセスから成る. MIDI データを入力, 出力するプロセスの定義は次のようである.

```
MIDIin::*[PortA?x; PortA?y; PortA?z; Transpose!(x,y,z)]
MIDIout::*[Transpose?(x,y,z);PortA!x; PortA!y; PortA!z]
```

PortA は特殊なデバイスで, 扱うのは (running status 無しの) note-on, note-off だけである. 移調プロセスの定義はこのようになる.

```
[MIDIin || Transpose || MIDIout]
```

このプログラムは PortA が閉じるまで動作し続ける.

MIDI Parser の例

MIDI 信号を受けとり構文解析する部分 (パーザ) を記述してみよう. 簡単のために running status, system exclusive 等は無視する.

```
Parser::
*[PortA?x; Channel := (x AND 0x0f); Status := (x AND 0xf0);
  [Status = NOTEOFF → PortA?Note; PortA?Vel; User!noteoff(Channel,Note,Vel)
  □
   Status = NOTEON → PortA?Note; PortA?Vel;
                     [Vel <> 0 → User!noteon(Channel,Note,Vel);
                     □
                      Vel = 0 → User!noteoff(Channel,Note,Vel);]
  □
   Status = POLYPRESS → PortA?Note; PortA?Press;
                        User!polypress(Channel,Note,Press)
  □
   Status = CONTROL → PortA?CtrlNum; PortA?CtrlVal;
                      User!control(Channel,CtrlNum,CtrlVal)
  □
   Status = PROGRAM → PortA?ProgNum; User!progchange(Channel,ProgNum)
  □
   Status = CHANPRESS → PortA?PressVal; User!chanpress(Channel,PressVal)
  □
   Status = PITCHBEND → PortA?BendLSB; PortA?BendMSB;
                        User!pitchbend(Channel,BendLSB,BendMSB)]
]
```

```
User::
*[Parser?noteoff(Channel,Note,Vel) → ... process noteoff ...
  □
  Parser?noteon(Channel,Note,Vel) → ... process noteon ...
  □
  Parser?polypress(Channel,Note,Press) → ... process polypressure ...
  □
  Parser?control(Channel,CtrlNum,CtrlVal) → ... process control change ...
  □
  Parser?progchange(Channel,ProgNum) → ... process program change ...
  □
  Parser?chanpress(Channel,PressVal) → ... process after touch ...
  □
  Parser?pitchbend(Channel,BendLSB,BendMSB) → ... process pitchbend ...
```

このパーザは, 生の MIDI 信号を受け取り (constructor で) パッケージ化し, クライアントのユーザプロセスに送信する.

音楽分析結果をプレイバックする例

音楽分析した結果に基づいて聴取プロセスをシミュレートするプログラムを記述しよう. 音楽分析としては, 簡単のため Lerdahl and Jackendoff, "A Generative Theory of Tonal Music" の特に time-span reduction (リズム的な各音の相対的重要さを階層構造の中で決定していく処理) だけ行う. 分析例を図 2.20 に示す. 我々のプログラムは, 分析の正しさを検証するために, 各階層を選択的に聴くことのできる分析家 (theorist) を摸擬する. 各階層は Level(i) という別々のプロセスで表現する. レベル i が activate されたら, その音の移調幅をレベル i-1 に送る. レベル 0 がトップレベルで, 図 2.20 のレベル a に対応する. レベル 0 は常にトニック 1 音だけから成る. プログラムは以下のようになる (% 以降はコメントである).

```
Level(0)::                          % トップレベルプロセス
[ Data(0)?(Dur,f);                  % tonic を読み込む
                                    % レベル 1 の状態を Run1 にセット
  Run1 := false; Level(1)?Running() → Run1 := true;
                                    % レベル 1 が run の状態ならデータを受け取り
                                    % 移調して出力
   [Run1 → *[Level(1)?T(Dur,TrPose); Score!(Dur,f*2^^(TrPose/12))]
   □
    not Run1 → Score!(Dur,f)]  % レベル 1 が run でないなら through
]

Level(i:1..4)::
[ Data(i)?Next;                     % Next に i-1 レベルの音の個数をセット
  j := 0; k := 0; l := 0;
  *[J < Next → Data(i)?Num(j); % Num(j) という配列要素に i-1 レベルの j 番目
                                    % の 1 音に対応する i レベルの音の個数をセット.
                                    % 音個数分だけ音価と移調幅を配列にセット
    *[l < Num(j) → Data(i)?(d(k), f(k)); k := k + 1; l := l + 1];
```

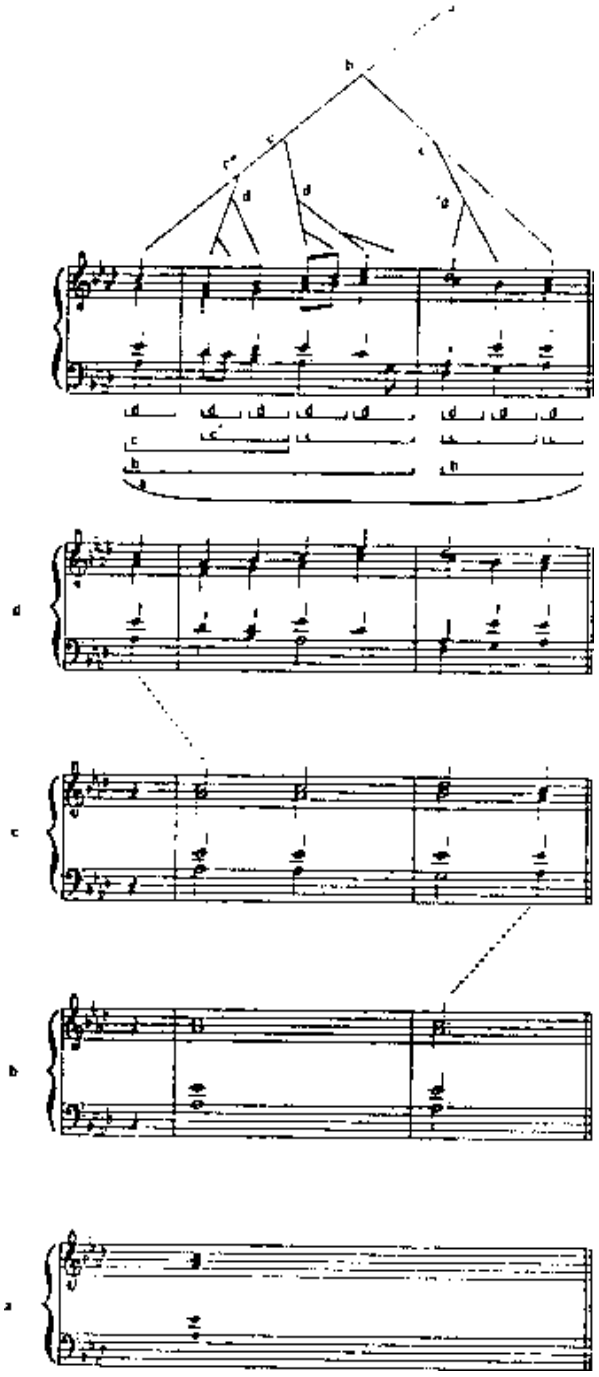


図 2.20: Time-span Reduction の例

```
  j := j + 1; l := 0];
Level(i-1)!Running();          % i-1 レベルに，レベル i が run していることを
                               % 知らせる
                               % レベル i+1 の状態を RunPrev にセット
RunPrev := false; Level(i+1)?Running() → RunPrev := true;
j := 0; k := 0; l := 0;
                               % レベル i+1 が run していれば:
                               % 聴取プロセスが次の音に移動する
```

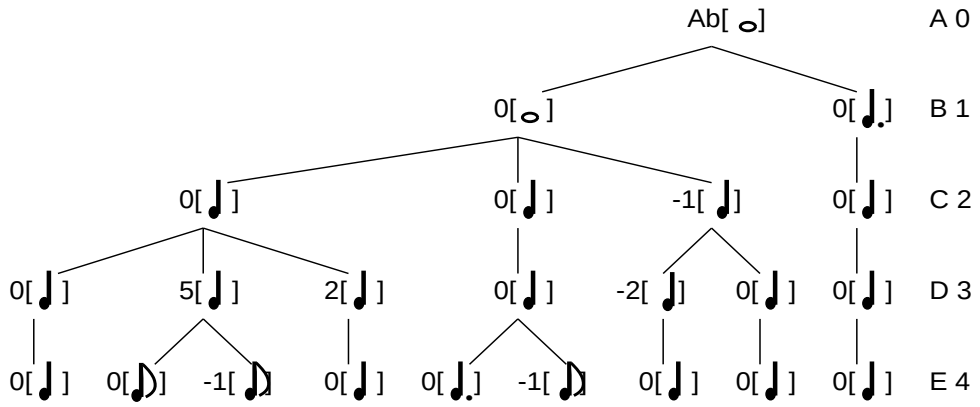


図 2.21: 分析データの階層構造

```
  [RunPrev → *[Level(i+1)?NewNote() → k := k + 1; l := l + 1
               □                    % 音価と（ひと固まり分の）移調幅を受けとる
                Level(i+1)?T(Dur,TrPose) →
                                    % そのひと固まりの音は全て聴き終わった
                  [l = Num(j) → Level(i-1)!NewNote(); l := 0; j := j + 1;
                   □
                   l < Num(j) → skip];
                                    % 各音毎に移調幅を加え合わせて i-1 レベルへ
                Level(i-1)!T(Dur,TrPose + f(k))]
  □                                 % i レベルが聴取プロセスの最下層である
   not RunPrev → *[j < Next → % Data プロセスから読んだデータをそのまま
                                    % i-1 レベルへ出力
                    *[l < Num(j) → Level(i-1)!T(d(k),f(k));
                                    k := k + 1; l := l + 1];
                    j < Next - 1 → Level(i-1)!NewNote();
                                    j := j + 1; l := 0]
  ]
]
```

プロセスを配列で表現したかったのでデータ部と分けて記述した．データ部は非常に大きいので一部分 (Level(0) と Level(2)) だけ挙げる．データ部の全体の構造を図 2.21 に示すので異なるレベルどうしがどのように関係しているか分かるであろう．図 2.20 の最低音に関するデータ記述は以下のようになる．

```
Data(0)::[Level(0)!(0N,Ab2)]
Data(2)::
[ Level(2)!2;                       % 直上のレベル（Level(1)）での音の個数
  Level(2)!3;                       % 第 1 レベルの第 1 番目の音に対応する
                                    % 第 2 レベルでの音の個数（= Num(0)）
    Level(2)!(2N,0);                % 第 2 レベルの 1 番目の音の情報
    Level(2)!(2N,0);
    Level(2)!(2N,-1);
```

```
  Level(2)!1;                          % 第 1 レベルの第 2 番目の音に対応する第 2
                                       % 第 2 レベルでの音の個数 (= Num(1))
    Level(2)!(4N,0);
]
```

二つ組の最初の数は音価 (duration) で, 1N は全音符, 2N は 2 分音符という具合である. 2 番目の数は移調の幅を半音単位で数えたものである.

例えばレベル 3 で聴取する状態 (図 2.20 で言えば d) は,

```
[Level(i:0..3) || Data(i:0..3)].
```

でシミュレートできる. 実際の人間の聴取過程は勿論かなり複雑であるが, この CSP によって記述されたモデルがその適切な出発点になっているであろう.

2.2.6 論理型プログラミング言語

述語論理を用いた研究は近年盛んになってきている. Ebcioğlu は BSL という言語を設計しバッハのコラールに関する知識を記述し, それから生成をおこなった [Ebc92]. Widmer は Lerdahl&Jackendoff の理論を Prolog で実装している [Wid92b] [Wid92a]. やはり Narmour の理論を Prolog で実装している研究もある [SH92]. 平田と青柳は Prolog で記述した和声理論を帰納的に学習することを試みている [HA92].

2.3 標準化について (長嶋, 松島)

演奏情報の標準的な規格については, MIDI (Musical Information Degital Interface) があり、一般的に広く使用されている。また SMF (Standard MIDI Files) は、MIDI 形式の演奏情報を記述した共通のファイル形式である。SMF については、MIDI 規格協議会からオフィシャルドキュメント (RP-001 Standard MIDI Files 1.0) が入手できる。これは Recommended Practice ということで、MIDI 規格そのものではない。この SMF ドキュメント入手手続きについてはまだ整備されていないが、問い合わせ先は下記である:

MIDI 規格協議会事務局
〒 136 東京都江戸川区亀戸 1-35-12 サンコモンズ 201
Tel: 03-3636-8908
Fax: 03-3636-9372

GM(General MIDI) は、MIDI 対応音源の音色配置や MIDI コントロールのプロトコルを共通化するための規格 (案) であり、これも MIDI 規格そのものではない。まだ整備されたドキュメントは用意されていないが、問い合わせ先は同じ上記 MIDI 規格協議会である。

また、MIDI 規格書の著作者は MIDI 規格協議会であり、MIDI 1.0 規格 (Document Ver 4.1 日本語版) の取次所は以下のところである。

株式会社 リット - ミュ - ジック 商品管理部
〒 160 東京都新宿区新宿 6 丁目 12 番 5 号 新宿松喜ビル
Tel: 03-3359-0267

音楽情報全般に亘る標準化の動きとしては, ANSI (American National Standards Institute), MIPS (Music Information Processing Stadards) 委員会で, 1986 年より音楽表記の標準化案作

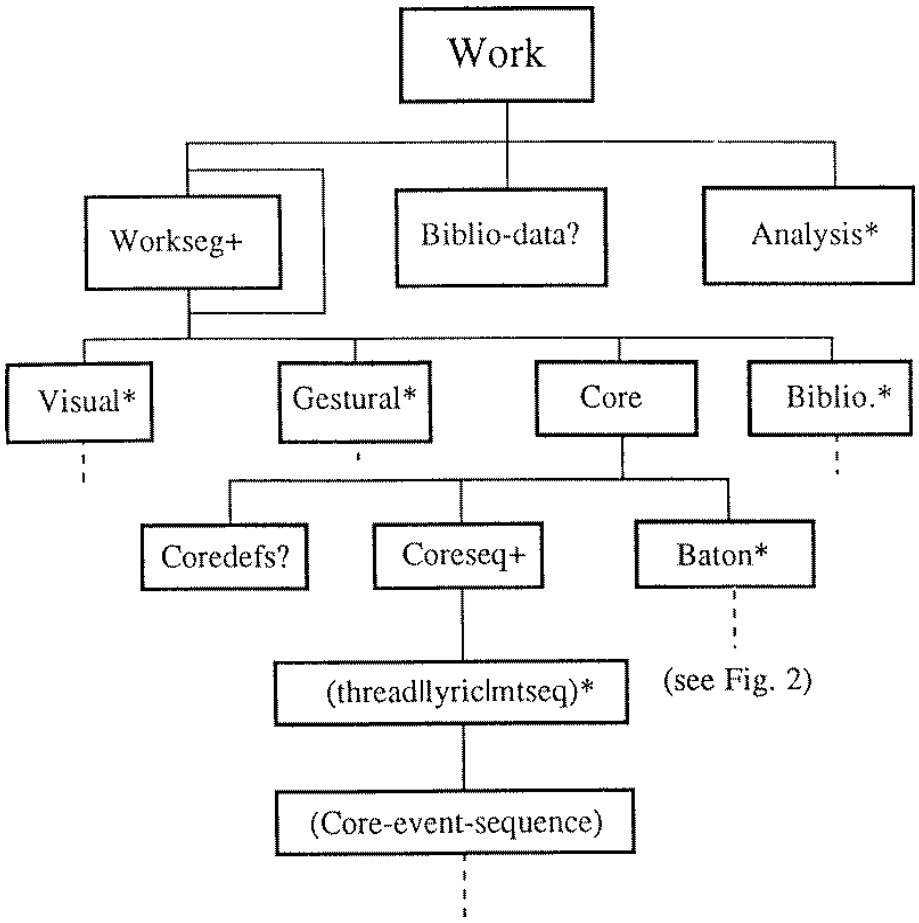


図 2.22: SMDL の構造

成を進め, 国際標準 (ISO standard) を目指した活動を行っている. この言語は SMDL (Standard Music Description Language) と呼ばれ, 1992 年 10 月現在の最新版は文献 [ISO91] である.

図 2.22 は, SMDL の上位レベル構造を表したものである [Slo89]. 各箱は要素 (element) を表し, これらが基本的な構造単位となる. SMDL で記述された文書 (document) は work と呼ばれる. しかしながら, work には完全な音楽作品が含まれるわけではなく, 一つのユニットとして表したいデータの集まり (作品の一部分やそれらの組み合わせ) が含まれる. work には, 書誌情報 (bibliographic data) や作曲家による解析 (analysis) が複数含まれても良い. また work は複数の work segment に分割しても良い.

work の中は, core, gestural, visual, analytical に分離して記述する. core には work の核になる情報 (全ての楽譜・演奏に共通な情報) が記述され, gestural には演奏固有の情報, visual には楽譜固有の情報が記述される. 記述される内容の例を表 2.6に示す [Slo89]. 当初, core の記述も SMDL と同じ規格で規程していたが, 最近, core については, 別の規格 "HyTime (Hypermedia/Time-based Document Language)" で記述することになった (図 2.23)[ANS90]. HyTime は, ハイパーテキストと時間記述能力を備えた記述言語で, 同じく ANSI の標準化案となっている [ISO].

SMDL と HyTime は, SGML (Standard Generalized Markup Language)[ISO86] により記述することが決まっている. SGML は, 文書 (document) 記述用のメタ言語で, ISO の国際標準となっている. SMDL に関する公式事務局は ICMA であり, 問い合わせ先は下記である:

International Computer Music Association, Inc.
2040 Polk St., Suite 330

表 2.6: SMDL で記述される内容

Object	Domain
note	core
clef	visual
tempo	core
chord symbol	core
Rubinstein's rubato of Chopin	gestural
Schenker's Analysis of Chopin	analytic
score layout	visual
fermata	core
Dorati's length of fermata	gestural
rhythm	core
time signature	visual
metric pulse	core
list of all editions	analytic
pitch	core
placement of note on staff	visual
spacing between staves	visual
trill	core
trill symbol	visual
Serkin playing a trill	gestural
discuss. of Serkin playing the piece	analytic

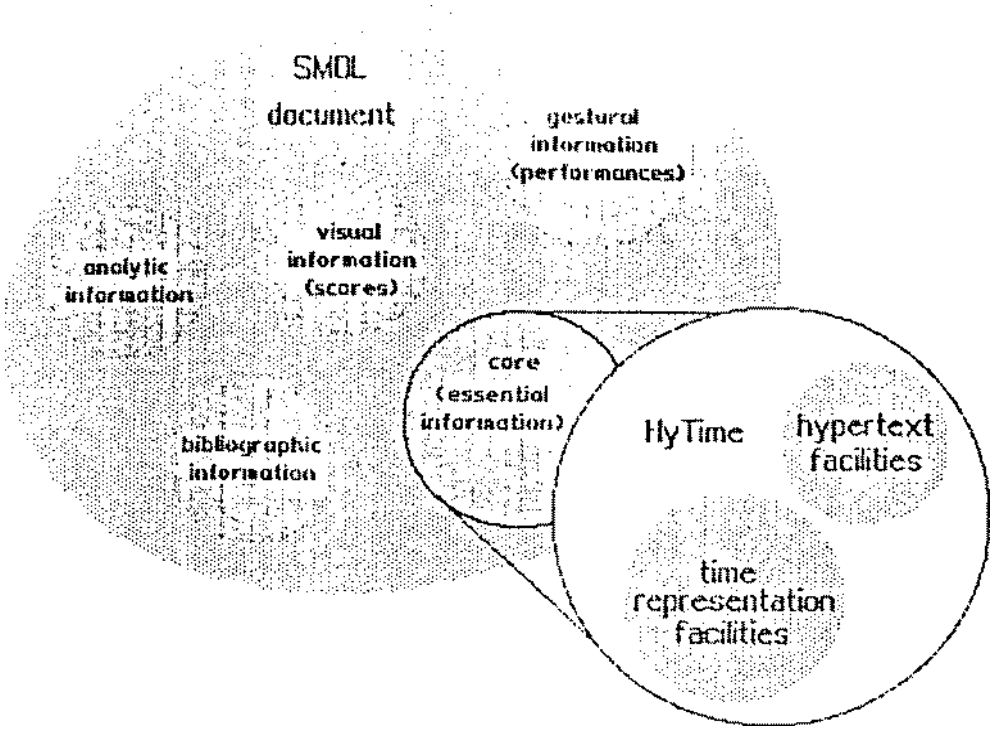


図 2.23: HyTime の位置付け

San Francisco, CA 94109
USA

参考文献

[Abb78a] Curtis Abbott. Machine tongues I. *CMJ*, Vol. 2, No. 1, pp. 4–5, 18, July 1978.

[Abb78b] Curtis Abbott. Machine tongues II. *CMJ*, Vol. 2, No. 2, pp. 4–6, September 1978.

[Abb78c] Curtis Abbott. Machine tongues III. *CMJ*, Vol. 2, No. 3, pp. 7–10, December 1978.

[Abb79] Curtis Abbott. Machine tongues V. *CMJ*, Vol. 3, No. 2, pp. 6–11, June 1979.

[ANS90] ANSI. *Hypermedia/Time-based Document Language "HyTime"*, 1990. X3V1/90–20.

[Bal86] M. Balaban. The tts language for music description. Technical Report 86–18, State University of New York at Albany, July 1986.

[BM70] S. Bauer-Mendelberg. The ford-columbia input language. *Musicology and the Computer*, pp. 48–52, 1970.

[Boy92] Lee Richard Boynton. Musical programming in scheme. In *Proceedings of ICMC 1992*, pp. 239–242, 1992.

[Bri86] A.R. Brinkman. Representing musical scores for computer analysis. *Journal of Music Theory*, Vol. 30, No. 2, pp. 255–276, 1986.

[Bro70] B.S. Brook. The plain and easie code. *Musicology and the Computer*, pp. 53–56, 1970.

[Che92] Marc Chemillier. Automata and music. In *Proceedings of ICMC 1992*, pp. 370–371, 1992.

[Dan89] Roger B. Dannenberg. The canon score language. *CMJ*, Vol. 13, No. 1, pp. 47–56, Spring 1989.

[DdP82] G.B. Debiasi and G.G. de Poli. Musica (musicae usitata scriptura idonee calculatoribus aptata): A language for the transcription of musical text for computers. *Interface*, Vol. 11, No. 1, pp. 1–27, 1982.

[Dol89] Mark Dolson. Machine tongues XII: Neural networks. *CMJ*, Vol. 13, No. 3, pp. 28–40, Fall 1989.

[DP76] G. De Poli. *MUSICA, un programme de codege de la musique, manual d'utilisation.* IRCAM, 1976. Paris.

[Ebc92] Kemal Ebcioğlue. *An Expert System for Harmonizing Chorales in the Syle of J. S. Bach*, pp. 294–333. The MIT Press, 1992. Understanding Music with AI: Perspectivews on Music Cognition, M. Balaban and K. Ebcioğlue and O. Laske (eds).

[Eri75] R.F. Erickson. The darms project: A status report. *Computer and Humanity*, Vol. 9, pp. 291–298, 1975.

[Eri76] R. Erickson. *DARMS: A Reference Manual.* Dept. of Music, Queens College, City Univ. of New York, June 1976.

[Fic91] Susan Fichera. Machine tongues XIII: Real-time audio conversation under a time-sharing operating system. *CMJ*, Vol. 15, No. 3, pp. 27–40, Fall 1991.

[Fox87] E. Foxley. *music: a musical preprocessor for troff.* ANSI, 1987. X3V1.8M/87-4.

[GL70] M.J. Gould and G.W. Logemann. Alma: Alphameric language for musical analysis. *Musicology and the Computer*, pp. 57–90, 1970.

[Gou86] J.S. Gourlay. A language for music printing. *Comm. ACM*, Vol. 29, No. 5, pp. 388–401, 1986.

[HA92] Keiji Hirata and Tatsuya Aoyagi. Towards inductive learning of jazz harmony theory. In *Proceedings of ICMC 1992*, pp. 46–49, 1992.

[Hir89] Yuzuru Hiraga. A computational model of the cognition of melodic/harmonic progression. In *Proceedings of ICMPC 1989*, pp. 61–66, 1989.

[Hol92] Frode Holm. Machine tongues XIV: CSP – communicating sequential processes. *CMJ*, Vol. 16, No. 1, pp. 25–33, Spring 1992.

[ISO] ISO/IEC. *Information Technology — Hypermedia/Time-Based Structuring Language (HyTime).* CD 10744.

[ISO86] ISO. *Information processing — Text and office system — Standard Generalized Markup Language (SGML)*, 1986. ISO 8879.

[ISO91] ISO/IEC. *COMMITTEE DRAFT, INTERNATINAL STANDARD, Information Technology — Standard Music Description Language (SMDL)*, 1991. CD 10743.

[Kor80] William A. Kornfeld. Machine tongues VII. *CMJ*, Vol. 4, No. 2, pp. 6–12, Summer 1980.

[Kra80] Glenn Krasner. Machine tongues VIII: The design of a Smalltalk music system. *CMJ*, Vol. 4, No. 4, pp. 4–14, Winter 1980.

[LdC91] Eric Lindemann and Maurizio de Cecco. Animal: Graphical data definition and manipulation in real time. *CMJ*, Vol. 15, No. 3, pp. 78–100, Fall 1991.

[Lev81] David Levitt. A melody description system for jazz improvisation. Master's thesis, MIT Department of Electrical Engineering, 1981.

[Lev84] David Levitt. Machine tongues X: Constraint languages. *CMJ*, Vol. 8, No. 1, pp. 9–21, Spring 1984.

[Lie82] Henry Lieberman. Machine tongues IX: Object-oriented programming. *CMJ*, Vol. 6, No. 3, pp. 8–21, Fall 1982.

[NF91] Rupert C. Nieberle and Markus Freericks. Extending an interactive music environment. *CMJ*, Vol. 15, No. 2, pp. 41–48, Summer 1991.

[Pop87] Stephen Travis Pope. A smalltalk-80-based music toolkit. In *Proceedings of ICMC 1987*, pp. 166–173, 1987.

[Pop89] Stephen Travis Pope. Machine tongues XI: Object-oriented software design. *CMJ*, Vol. 13, No. 2, pp. 9–22, Summer 1989.

[Pop92a] Stephen Travis Pope. The interim dynapiano: An integrated computer tool and instrument for composers. *CMJ*, Vol. 16, No. 3, pp. 73–91, Fall 1992. also available through anonymous FTP from ccrma-ftp.stanford.edu.

[Pop92b] Stephen Travis Pope. The SmOKe music representation, description language, and interchange format. In *Proceedings of ICMC 1992*, pp. 106–109, 1992. also available through anonymous FTP from ccrma-ftp.stanford.edu.

[Puc91] Miller Puckette. Combining event and signal processing in the MAX graphical programming environment. *CMJ*, Vol. 15, No. 3, pp. 68–77, Fall 1991.

[Rah90] John Rahn. The Lisp Kernel: A portable software environment for composition. *CMJ*, Vol. 14, No. 4, pp. 42–58, Winter 1990.

[Roa79a] C. Roads. Machine tongues IV. *CMJ*, Vol. 3, No. 1, pp. 8–13, March 1979.

[Roa79b] C. Roads. Machine tongues VI. *CMJ*, Vol. 3, No. 4, pp. 6–8, December 1979.

[Rob67] T.D. Robison. Iml–mir: A data-processing system for the analysis of music. In H. Heckmann, editor, *Elektronische Datenverarbeitung in der Musikwissenschaft, Regensburg*, pp. 103–135. Gustav Bosse Verlag, 1967.

[Row92] Robert Rowe. Machine listening and composing with Cypher. *CMJ*, Vol. 16, No. 1, pp. 43–63, Spring 1992.

[Sca89] Carla Scaletti. The Kyma/Platypus computer music workstation. *CMJ*, Vol. 13, No. 2, pp. 23–38, Summer 1989.

[Sch91] Will Schottstaedt. Common Lisp Music. available through anonymous FTP from ccrma-ftp.stanford.edu, 1991.

[SH92] Matt Smith and Simon Holland. An AI tool for the analysis and generation of melodies. In *Proceedings of ICMC 1992*, pp. 61–64, 1992.

[Slo89] Donald Sloan. Precis of the standard music description language. In *Proceedings of 1989 ICMC*, pp. 296–302, 1989.

[TA91] C.P. Tsang and M. Aitken. Harmonizing music as a discipline of constraint logic programming. In *Proceedings of ICMC 1991*, pp. 61–64, 1991.

[Tal87] A.D. Talbot. *Journal of Technical Development*. ANSI, 1987. X3V1.8M/87-35.

[Tau91] Heinrich Taube. Common Music: A music composition language in Common Lisp and CLOS. *CMJ*, Vol. 15, No. 2, pp. 21–32, Summer 1991.

[Tod88] P.M. Todd. A Sequential Network Design for Musical Applications. In *the 1988 Connectionist Models Summer School, San Mateo CA*. Morgan Kaufmann, Pulishers, 1988.

[Tod92] Peter M. Todd. A connectionist system for exploring melody space. In *Proceedings of ICMC 1992*, pp. 65–68, 1992.

[Wen70] J. Wenker. A computer oriented music notation including ethnomusicological symbols. *Musicology and the Computer*, pp. 91–129, 1970.

[Wen72] J. Wenker. A music compiler for musicologist and ethnomusicologists, an analytical approach to folk music, and mustran ii – An Extended MUSic TRANslator. *Computer and Humanity*, Vol. 7, No. 2, pp. 124–126, 1972.

[Wid92a] Gerhard Widmer. *A Knowledge Intensive Approach to Machine Learning in Music*, pp. 490–507. The MIT Press, 1992. Understanding Music with AI: Perspectivews on Music Cognition, M. Balaban and K. Ebcioğlue and O. Laske (eds).

[Wid92b] Gerhard Widmer. Qualitative perception modeling and intelligent musical learning. *CMJ*, Vol. 16, No. 2, pp. 51–68, Summer 1992.

[WZ88] R.J. Williams and D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1988.

[松島 88] 松島俊明, 神前尚生. 音楽表記の標準化動向. 情報処理学会誌, Vol. 29, No. 6, pp. 566–578, 1988.

第 3 章

音楽認識

本章では、パターン認識という観点から取り組まれてきた音楽情報処理研究についてサーベイを行なう。

従来より音楽分野においてはデータを計算機に入力する手段が待望されていた。MIDI 楽器の登場により音楽の入力作業は大幅に軽減されたが、依然として楽譜、音響といったパターン情報からのデータ入力に対するニーズは高い。第 1 節においては楽譜から音楽データを入力する手段、すなわち楽譜認識について述べる。ここでは、今までの研究を手法・対象別に整理し、さらに楽譜認識の過程に添ってサーベイを行なっている。第 2 節では、音響からデータを入力することを目的とした自動採譜とについて述べる。ここでは、音響信号から楽音への群化、楽音の音楽的な解析、楽譜化という流れでサーベイを行なっている。第 3 節では演奏表現の生成について紹介する。生成という過程は一見認識と関係ないように思われるが、生成のモデル化の際には解析系の研究が中心課題となっている。音楽情報処理分野の中でも重要な分野であるため本章で取り上げている。第 4 節では近年注目を集めている感性情報の処理について述べる。コード、リズムなど、さらにはそれらに込められた意図・情緒の認識はヒューマンコミュニケーションにおいて不可欠である。ここでは Cypher というシステムを中心に感性情報の処理について述べる。

3.1 楽譜認識 (松島)

音楽情報の計算機入力の一方法として, 楽譜認識 (読譜) がある. 楽譜認識の研究は, 音楽情報処理の分野のなかでも日本における研究が活発な分野で, 研究例の多くは国内における研究である.

この研究の報告は比較的古くからあるが, 研究例はそれほど多くない. MIT の Do-Re-Mi[Pre71], 電総研 [青山 82a], 東京大学 [尾上 79], 大阪大学 [中村 79][加藤 86a][加藤 86b], 早稲田大学 [松島 85], McGill 大学 [FAPH91], 長岡技大 [宮尾 92] などで研究例があるが, 初期の研究のほとんどは処理速度や認識できる記号の種類などに問題があり, 実用化には至っていない.

対象とする楽譜は, 西洋の標準的記譜法により記述された市販の印刷楽譜 (いわゆる五線譜) であり, また, ピアノ楽譜が取り上げられることが多い. 難易度 (画像としての複雑さ) にも大きく開きがあり, 初期の研究では, 単音かつ五線一つ分という単純な構成のものを対象としていたが, 近年では和音や複数旋律の楽譜を対象とするようになってきた. また, 楽譜に含まれる記号は多種にわたるため, 簡単な楽譜に対しては, 音符や休符など基本的記号を高認識率で認識可能な報告も出ているが, 楽譜中のほとんど全ての記号を認識できるような実用に耐え得る楽譜自動認識は未だない. 表 3.1 に主な研究例を示す.

楽譜認識では, 楽譜中に表記される記号の呼称は一般的な呼称を使うが, 音符については音符の各部を呼び分ける必要がある. 本節では, 以下, 次の用語を使って説明する (図 3.1):

- 符頭 (Note Head): 音符の玉の部分. 黒符頭, 白符頭, 全音符の 3 種類.

表 3.1: 楽譜認識研究例

研究機関	MIT	阪大 1	電総研	早大	阪大 2	McGill 大	長岡技科大
論文発表年	1971[*1]	1979[*2]	1982[*3]	1983[*4]	1986[*5]	1989[*6]	1990[*7]
目的	データベースの作成	データベースの作成	データベースの作成	実時間演奏[*8]	データベースの作成	データベースの作成	データベースの作成
楽譜の種類[*9]	単音 単旋律	単音 単旋律	和音 複旋律	和音 単旋律	和音 複旋律	和音 複旋律	和音 複旋律
認識対象記号[*10]	基本記号	基本記号	基本記号 他への拡張性への言及有	基本記号 スタッカート, テヌート等	基本記号	基本記号 他は不明	基本記号と スラー, タイ, クレッシェンド フォルテなど
認識率[*11] (最も低い例)	不明	不明	不明	98% (こぎつね)	83.1% (悲愴 第 2 楽章)	不明	93.5% (別れの曲)
認識時間	2-3 小節で 4 分	不明	不明	10 秒前後	平均 90 分	不明	19 分
主な認識方法[*12]	外接四角形による分類	プロジェクションによる分類	外接四角形による分類と構造解析	パターンマッチングと周辺探索	黒板モデルによる仮説検証	モーメント特徴と学習	外接四角形による分類と周辺探索

***1** [Pre71] [Pre75]

***2** [中村 79]

***3** [青山 82a] [青山 82b] [青山 84b] [青山 84a]

***4** [園元 83] [小沼 85] [園元 84a] [園元 84b] [原田 84] [園元 85] [松島 85] [MHS$^+$85] [二村 87]

***5** [加藤 86a] [加藤 86b] [加藤 88]

***6** [FPA89] [FAPH91] [FAPD92]

***7** [宮尾 89] [宮尾 90a] [宮尾 90b] [宮尾 92]

***8** 楽譜を数秒で読み取り, 直ちに演奏する. データベース作成目的の場合, 認識誤りを人手により修正することが可能であるが, 演奏目的では, 認識誤りがあった場合, 音楽的に矛盾がないよう自動的修正を行う必要がある.

***9** 1 つの符尾に符頭が 1 つだけ付くものを対象としているものを単音, 複数の符頭を許すものを和音で表した. プロジェクションのような簡単な特徴のみでは和音の認識は難しい. また, 1 つの五線上に旋律が 1 つだけ書かれているものを単旋律, 複数の旋律を許すものを複旋律とした. 複旋律の場合, 小節内の拍数の検査は旋律毎に行わなければならないが, 阪大 2 はこれを考慮している. 表に挙げた例は, 全て伴奏付きのピアノ曲のように複数のパートからなる楽譜を対象としている.

***10** ここでいう基本記号とは, 音符, 休符, 付点, 臨時記号, 音部記号, 拍子記号, 調号, 小節線等, 音の高さと長さを表すのに最低限必要な基本となる記号であるが, 各研究により若干の差異がある.

***11** 楽譜の認識率の定義は各研究でまちまちである. 早稲田では, 認識結果を演奏した時に, 元の楽譜と同じ音が出ている時間の比で求めている. 例えば, 音部記号の認識を誤って場合, 全ての音符を正しく認識しても, 認識率は 0 となる. 阪大 2 では, データベース化する際に人間が修正しなければならない音と数の割合で表している. 長岡技科大は (正しく認識された記号数)/(全記号数) で求めている.

***12** 各研究で特徴としている方法を示した. 過去の研究成果を取り入れている部分もある. 詳細は第 3.1.4節で述べる.

- 符尾 (Stem): 2 分音符以下の音符に付く棒.
- 符鉤 (あるいは単鉤) (Hook): 8 分音符以下の音符に付く旗.
- 連桁 (Beam): 8 分音符以下の音符を複数つなげる旗.

図 3.1: 音符各部の用語の説明

3.1.1 楽譜認識の問題点

楽譜は, 画像認識の観点からみれば, 白と黒の二値画像であり, また, 出現する記号もそのほとんどが直線や楕円などの幾何学図形からなるため, 一見その認識は容易そうである. しかし, 実際に試みると様々な問題に直面し, 実用に耐え得るものを作成するのは容易ではない.

楽譜を主として自動認識の立場から見てみると, 以下のような点に留意する必要がある.

1. 高分解能での画像入力
   楽譜は五線や小節線のような細長い直線成分, 付点やスタッカートのような微小な記号等, 微細な構造の記号を含む二値画像である. したがって, 細かい直線の切れ・かすれを防ぎ, 高品質の画像を得るためには高分解能での入力がが必要となる.
2. 認識対象外記号の扱い
   楽譜特有の記号の他に日本語, 外国語, 数字などが入り交じって含まれており, しかも出現位置の特定も困難である. これらの記号を楽譜記号とは区別して認識, あるいは無視することが必要となる.
3. 楽譜記号の多様性
   活字のように変形しない記号 (符頭・音部記号など) と変形するが意味的に同じ記号 (符尾・タイ・スラー・連桁など) が存在する. また, 変形しない記号でも, 楽譜により微妙にその形状が異なることがある. このためパターンマッチングによる認識と構造解析的な認識とを併用する必要がある.
4. 楽譜の難易度差
   音符密度が低く, 限られた数の基本的な記号しか表記されない平易な楽譜から, 音符が密集し, 様々が記号が書き込まれている難しい楽譜まで, 楽譜自体の難易度差 (演奏の難易度とは必ずしも一致しない) が大きく, 対象とする楽譜のレベルを考慮する必要がある. また, 1 曲が数枚にわたるような楽譜は, 頁毎に独立ではないため, 頁間の意味的なつながりを考慮する必要がある.

また, 従来の楽譜自動認識研究の問題点について, [CBT92] は以下のように指摘している.

1. 認識率
   従来の文献の認識率はいずれもほぼ 100% の認識率を得たと報告しているが, それらはいずれも非常に少ないサンプルについのものであり, より多くの楽譜について実験を行えば, さまざまな問題に直面するであろう.
2. 記号の融合
   個々の記号を認識するためには, まず, 各記号を他の記号と分離する必要があるが, 楽譜はある意味で記号の配置は適当 (人間の気まぐれにより適当に配置されてしまう) なため, 異なる記号が接近もしくは連結されていることが度々生じる.

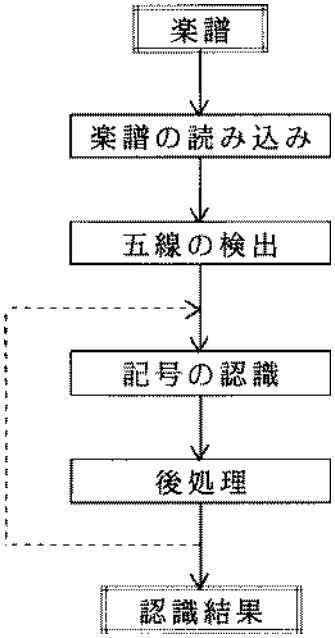


図 3.2: 楽譜認識の流れ

3. 印刷ミス
   人間にとっては容易に正しい記号を判別できるような印刷ミスでも, パターン認識の方法では難しいことが多い. これら印刷ミスは, インクが不適当な場所についていたり, 印刷時のインクの載りが完全でなかったような場合に生じ, 極端なときには黒符頭が白符頭になってしまう場合もある.

これらの問題点は, 楽譜版下の作成は主に手書きで作成されており, 一つの芸術作品であるという楽譜の作成方法に由来するものが多い. さらに, 印刷されたものだけではなく, 手書き楽譜の認識はいっそう困難である.

以下では今までの研究例に基づき, 楽譜認識の基本的な手法について解説する. 各文献では, 細部についてはそれぞれ特有の方法により認識が行われているが, 大まかな認識の流れや認識方法には共通する部分が多い. 以下, 画像の入力, 五線の検出, 記号の認識, 認識結果の検証・使用目的に合わせた形式への変換などの後処理の順に述べる (図 3.2).

3.1.2 画像の読み込み

楽譜は付点のような小さな点や五線や小節線のような細線が重要な構成要素となっている. これらの微細な記号の認識を行うためには, 楽譜画像は高解像で入力する必要がある. そのため以前は画像の入力にフライングスポットスキャナなど大がかりな装置が必要であったが, 近年はフラットベット型のイメージスキャナで 240dpi 程度の解像度で読みとる. A4 サイズの楽譜 1 枚でほぼ 2000×3000 ドット (約 600 万画素) になる. A4 サイズの楽譜では, 付点や五線などの微細な記号はおおむね 0.2mm から 0.5mm 程度となるので, この解像度で五線の幅は 2 から 5 点になる.

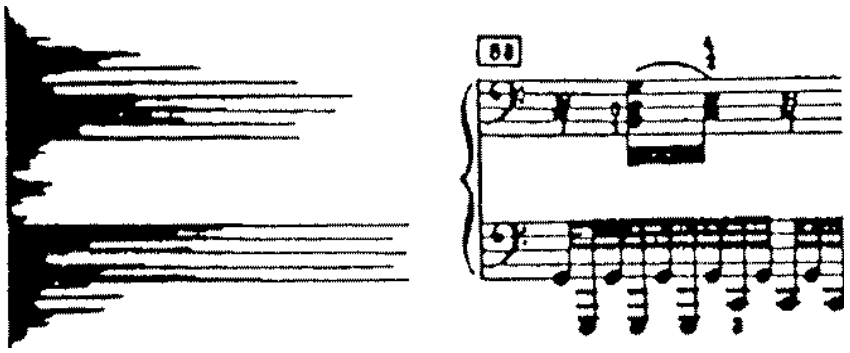


図 3.3: 周辺分布による五線の検出

3.1.3 五線の検出

楽譜は五線を基準として表されている．そのため, すべての記号の認識に先だってまず, 楽譜中から五線を検出する必要がある．五線は記号の位置情報の基準としてだけではなく, 記号の大きさの基準としても用いられる．即ち, 五線の線間が正規化の基準にとられる．五線 (stave) は等間隔に並んだ水平な五本の直線である．したがって, 検出にはその幾何学的性質 (直線性, 平行性, 等間隔性) が利用される．もっとも簡単な方法は水平方向の周辺分布 (Projection Profile) を調べ, そのピーク位置で五線の位置を検出する方法である (図 3.3)[FAPD92].

しかし, 上述のようにイメージスキャナ等で高解像で読みとられた楽譜画像においては, 五線の水平性, 直線性は必ずしも保証されないことが多い．そのため, 五線の歪み, 傾き, 切れ・かすれ, 長いスラーや水平な連桁の連続があっても五線を正しく検出できるような処理が必要となる．例えば, まず, 垂直方向に数カ所で粗走査を行い, 五線の候補を抽出する．その際, 一枚の楽譜上では五線間隔はほぼ一定であることを利用し, 白画素のラン長の分布から五線の線間隔を決定し, 五線の候補を決定する．各走査位置での五線候補を連結して五線位置を決定する (図 3.4)[青山 82a][MHS+85][加藤 88]. また, ハフ変換により五線を検出する試みもある [宮尾 92].

五線は他の記号の位置や意味を決定する重要な情報であるが, 一方, 他の記号との重なりが多く, 一度その位置が判別できれば, かえって邪魔な存在となる．したがって, 五線は検出された後に消去することが多い．五線を消去することにより, 他の記号が分離され, 以後の認識が容易になる．しかし, 単純に五線位置にある黒画素を消去すると, 本来つながっている記号を不用意に分離してしまう．そのため, 五線に沿って黒画素の幅を調べ, 線幅より広い場合には消去をしない方法 [青山 84b], 線幅より広い画素のつながりを消去してゆき, 最後に原画像との排他的論理和をとり五線だけを消去する方法などの工夫が行われている [FAPD92][加藤 88] (図 3.5).

3.1.4 記号の認識

前述のように, 楽譜認識はパターンマッチングに適する記号と, 構造解析的手法に適する記号とがあり, これらが相補的に利用されることが多い．今まで報告されている研究例はその方式により, 完全ではないが大きく 2 方式に分類できる．ひとつは連結した黒画素領域に対して大分類を行い, 各分類内で詳細な検査を行って認識する方法であり, もうひとつは楽譜記号をまず基本的な部分記号 (部品) に分割して考え, これら部分記号間の構造的特徴を利用して認識する方法である．前者は領域優先型の認識で, 例えば音部記号, スラー, クレッシェンドなどその大きさでほぼ判別可能な記号や, 出現位置が特定できない記号の認識に有利である．後者は, 形状優先型の認識で, 符頭, 符尾, 符鉤, 連桁等の要素の接続として考えることができる音符の認識に有利である．

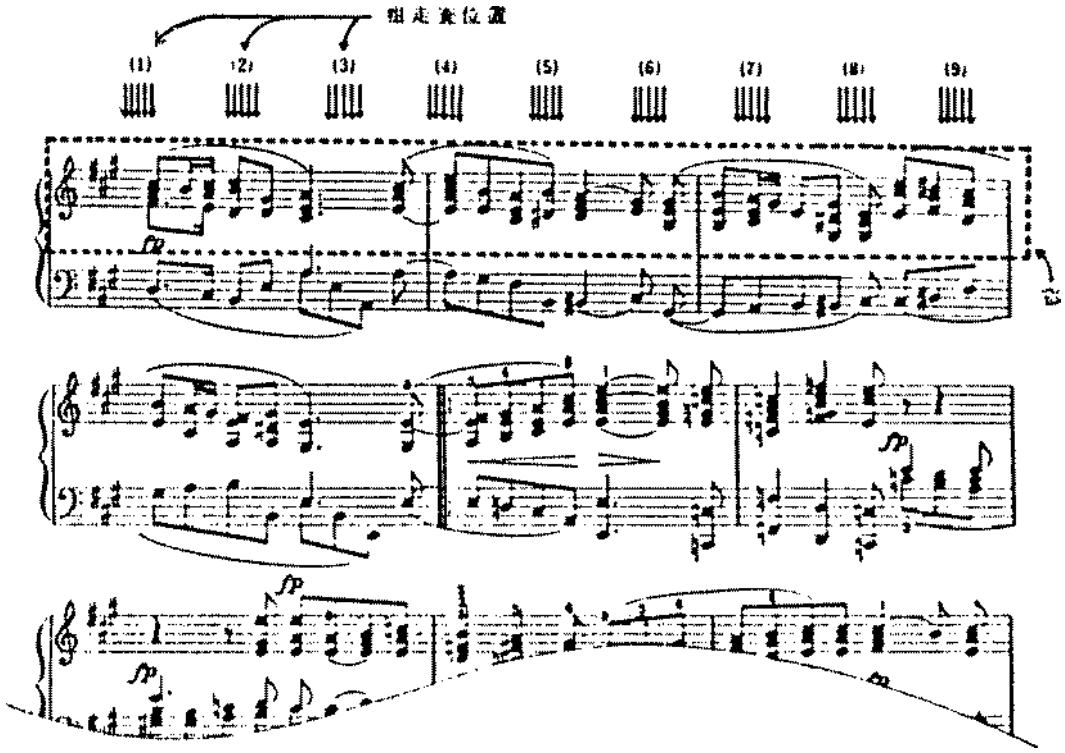


図 3.4: 垂直位置での走査による五線検出方法

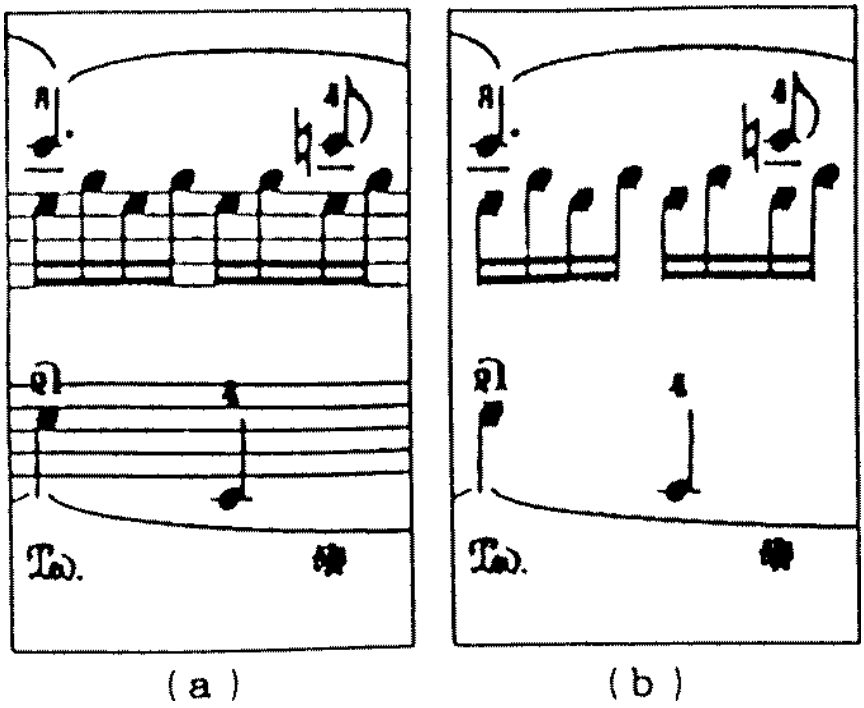


図 3.5: 五線の消去 ((a) 源画像 (b) 五線消去後の画像)

前者の方式は初期の研究に主として用いられた [Pre71][青山 82a][尾上 79][中村 79]. まず記号を幾つかのパターン・クラスに大分類したのち, それぞれのクラスに個別の識別処理を行い認識

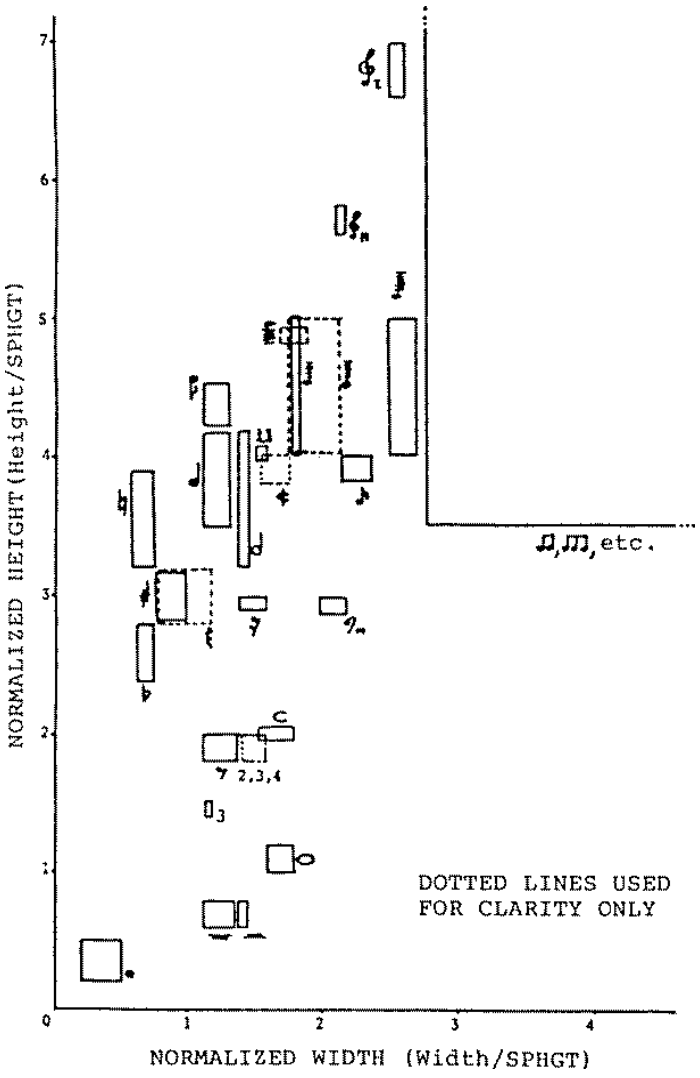


図 3.6: 大分類のための記号空間

する．大分類には，セグメントの外接四角形の大きさによる分類法や，プロジェクション・プロフィールの特徴パラメータ (ピーク数, ピーク位置, セグメント幅等) による分類法がある．

外接四角形による分類法では，図に示すように五線を単純消去した図形に対して外接四角形を考え，接続関係を考慮して記号全体の大きさにより記号空間上で種類分けし (図 3.6[Pre71])，楽典規則によるふるい落しと，外接四角形の大きさ・接続関係を特徴パラメータとする識別を試みている．また，五線消去の際に記号部分の欠落を防ぐアルゴリズムを用いて，分離した記号の外接四角形の大きさにより記号空間上で分類し，各グループ内の記号に対して縦横の黒ラン長を特徴パラメータとする個別の分類方法を適用した報告もある [青山 82a]．また，音符の場合，符頭と連桁 (符鉤) を分離してそれぞれを識別し，符頭と符尾の位置関係を解析して連桁の数を決定することも行われている [Pre71]．

プロジェクション・プロフィールによる分類法では，パターン内の特徴点の位置情報が重要な意味を持つことを利用して，垂直・水平方向のプロジェクションにより記号の位置情報と形状情報に関する特徴を抽出して認識を行う．垂直方向のプロジェクションによりセグメンテイションを行い，図 3.7 のように分離した領域に対して水平方向のプロジェクションを求め，ここで得られた特徴 (ピークの数・位置，下部・上部の幅，セグメントの幅等) により分類する [中村 79]．また，

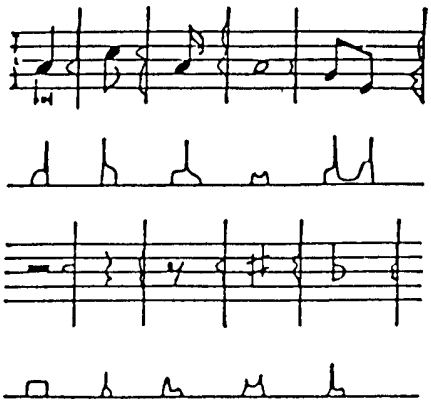


図 3.7: 周辺分布による記号の認識

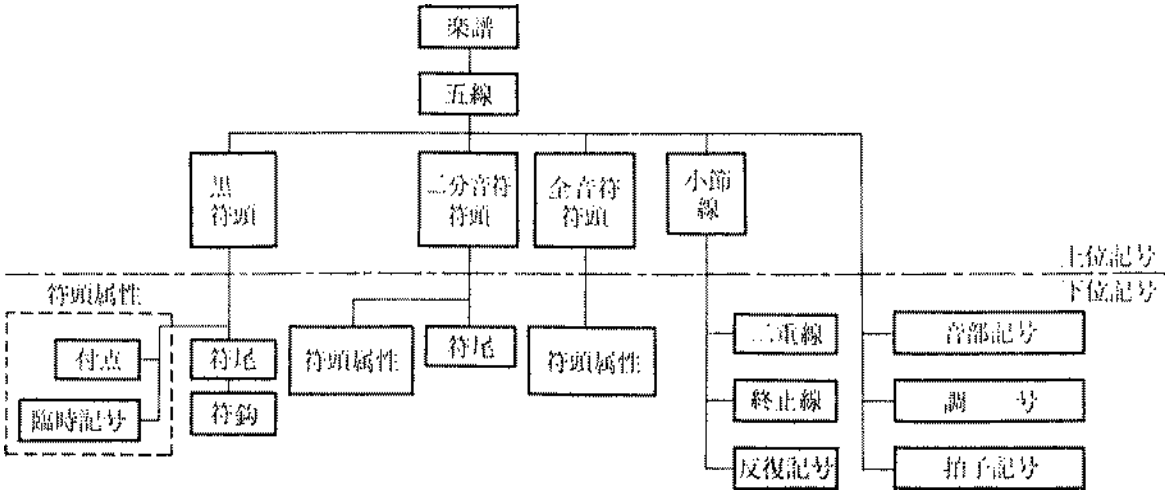


図 3.8: 楽譜記号の階層構造

垂直方向のプロジェクションによりセグメンテーションを行ったのち, ピークの数・位置で分類し, 更にピーク位置近傍の垂直ラインパターンの部分的マッチングをとる例もある [尾上 79].

後者による楽譜認識の例としては, 早稲田大学と大阪大学の研究例がある [松島 85][加藤 86a][加藤 86b]. 早稲田大学では, 楽譜記号を図 3.8 に示した階層構造で捉え, 上位に位置する記号はパターンマッチングによりその候補を検出したのち, 構造的特徴を利用して周辺探索し識別する方法を用いている. 検出には専用ハードウェアが用いられ, 五線, 小節線, 黒符頭, 白符頭の候補を高速に検出する. 黒符頭の音符を例に認識方法を説明すると, まず, 検出された符頭候補の右上隅および左下隅からそれぞれ上下方向に黒画素をたどり符尾の存在を確認する. 次に, 符尾の先端の右側および左側を符尾と平行に探索し, 符鉤および連桁の数を数える. 音符に対してその出現位置が特定できる記号 (付点やスタッカート等) は, 出現位置を探索して存在を調べる. このようにして認識された音符は, 小節単位で楽典検査を行い, 矛盾が生じた場合は修正を行う. この研究は, 実時間で楽譜を読み, エレクトーンを演奏するロボット: WABOT-2 の視覚系として筑波万博で 6 カ月間デモを行った.

大阪大学では, 楽譜記号を構成する部品を抽出し, 各部品を組合せて, 音楽的知識に合致したものを楽譜記号として認識している. 図 3.9[加藤 88] に示すように, 5 レベルの黒板, 4 つの処理

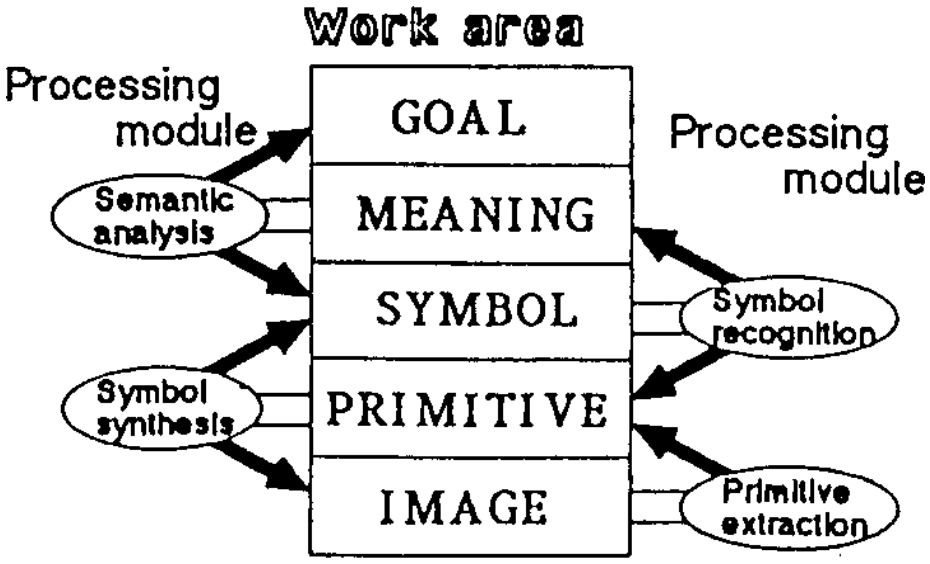


図 3.9: 黒板モデルによる楽譜認識

モジュール, 制御モジュール, 知識ベースからなり, 小節内での拍の合計が拍子記号で定められた拍数になるというゴールに向かって, 上位の処理モジュールが立てた仮説 (記号の候補) を, 下位の処理モジュールの認識結果が満足するまで, 上下モジュール間で仮説検定を繰り返しながら, シンボルの認識を行なっていく.

各処理モジュールは相互に干渉し, 仮説の変更や認識のやり直しを行う. このとき, 基本的・厳密な条件からより例外的・ゆるやかな条件へと認識条件を変更する.

以上の各方式を比較すると, 領域優先の方法は, 適当に選んだ特徴パラメータにより比較的簡単な分類アルゴリズムが採用できる. 特に外接四角形による分類では, 付点やト音記号のように大きさだけで記号を特定でき, 認識対象外記号でも分類アルゴリズムの拡張が比較的容易に行える. 反面, 識別以前の処理 (五線の消去・記号分離) がやや複雑で, 和音や連桁の記号分離に特殊処理を必要とし, ハードウェア化・高速化が困難である. またプロジェクションプロフィールによる分類法は簡単でよいが, 和音のように同一垂直位置の記号識別が困難であったり, 認識対象記号の拡張性にも問題がある. 一方, 形状優先の方法では, パターンマッチングに適し, 画像の広領域にわたる参照が必要な記号の検出を高速に行い, これを利用して記号の識別に必要な画像参照範囲を限定できるため, 高速化に適している. また, 必ずしも五線消去の必要がない. その反面, 探索場所を特定できない記号 (速度記号・クレッシェンド等) では, 上述の特徴を生かすことができない. 大阪大学の方式 [加藤 88] は, 仮説検証モデルの考え方を楽譜認識に導入したものであり, より正確な認識が可能である反面, 処理に時間がかかる.

最近の研究としては, 領域優先と形状優先の両方式の良い点を融合し, 五線除去による記号の大分類と周辺探索による音符認識により楽譜認識を試み, 高認識率の達成が報告されている [宮尾 92]. また, 誤認識をしたパターンのパラメータを学習により更新する例 [FAPD92], ニューラルネットを用いて手書き楽譜の認識を試みた例 [YBD+92] の報告もある.

3.1.5 後処理

認識された記号は, 演奏情報やデータベース用の形式に変換される. 最終的な楽典検査等が行われ, 認識結果が疑わしい場合は再認識に回されることもある. 前述したように記号の出現位置に対する制約も含まれるが,

- 音部記号は五線の最初に必ずある
- 調号は音部記号の直後か複縦線の後のみに書かれる
- 拍子記号は曲の最初に必ず書かれ, 複縦線の後に書かれることもある
- 符尾の両端に符鉤は付かない
- 単鉤には符尾が 1 本のみ付く
- 連桁には 2 本以上の符尾が付く
- 符尾の両端には符頭は付かない
- 符頭に付く符尾は上下各 1 本以下である
- 1 小節内の音符の拍の合計は拍子記号で定められた拍数になる
- 異なる旋律の対応する拍の音符は垂直方向にほぼ同一の位置に書かれる

などの性質が, 認識や後処理に積極的に利用されている.

3.1.6 市販システム

楽譜自動認識はそのほとんどが大学等での研究に限られているが, 筆者の知る限り唯一の市販楽譜読み取りシステムに「よみとりくん」がある. ローランドの協力によりジェー・イー・エルが開発し, 日本ソフトバンクから平成元年に販売された (48,000 円). 楽譜の入力はハンディースキャナで行う. 読み取り範囲は幅 32mm× 長さ 240mm で, 一度に五線 1 段分が入力できる. 認識できる記号は, 全音符から 16 分音符までの音符, 休符で, 誤認識した記号や調号, 拍子記号, 音部記号, 臨時記号などの認識対象外の情報・記号は付属のエディタなどにより手入力するなど, その機能は限られてものであった. 認識結果は「ミュージくん」で演奏出力ができる.

3.2 採譜 (片寄)

楽譜認識システムが視覚からのデータの収集を目的にしているのに対し, 聴覚を通して音楽データを収集する目的をもっているのが自動採譜システムである.

自動採譜のタスクは, 楽音への群化問題, 音楽的分析, 楽譜出力に大きく分けることができる. 楽音への群化問題とは一つの楽音を一つの楽音として聞く機能であり, 処理としては主として基本周波数の認識と考えることができる. 音楽的分析とはリズム・拍子認識, 楽音の音階への割当など音楽的な解析を伴った分析である. このような分析を行なった上で, さらにシンボルのグループ化, 位置決めなど記譜に関する処理を行なった上で採譜という処理が完結する. (ここでは, 伝統的な西洋音楽を対象として処理を考える. 音楽には各種の民族音楽・コンピュータミュージック形態の異なった音楽が多数存在し, それに合わせ, 複数の楽譜形態が存在するのは言うまでもない.)

また, 楽譜化というところからは少し離れるがリアルタイムで生音から音楽情報 (音高, 時間等) を抽出することから, 自動伴奏, インタラクティブコンポージングにおける制御信号としても利用されている.

3.2.1 楽音への群化

信号処理手法は基本的には 1 章, 音響信号処理の章で詳しく述べられているので, ここでは, 大まかに採譜処理で使われる周波数解析手法の紹介を行なう.

一般的に周波数抽出法としては, 入力信号とその時間遅れ関数の相関関係から求められる平均振幅差関数 [Moo74] があるいはその逆数的な意味合いをもつ GS D がハードウエア的にも実現しやすい手法として知られている. これらの手法は特にモノフォニーを対象によく用いられてきた. その他に, DFT や FFT, 線形予測バンドパスフィルタアレイなどによる周波数分析が採譜処理に

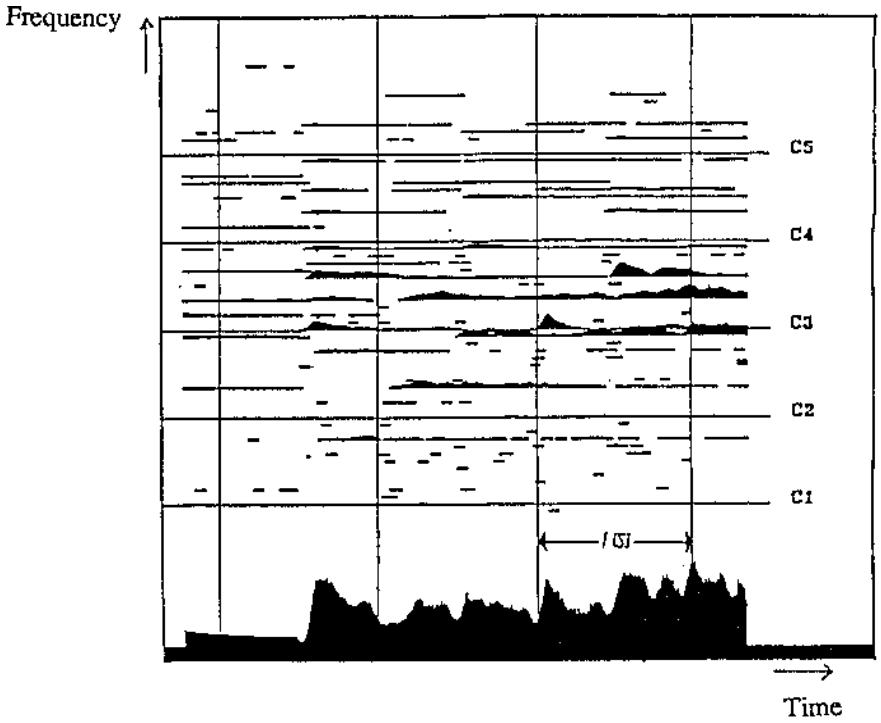


図 3.10: 時間周波数マップ: 複素スペルトルの内挿法を用いてピアノ音を分析したもの

用いられている. 中でも FFT は高速である, パッケージ化が進んでいるなどのメリットがあったため多くの採譜システムに利用されてきた. FFT は高音部においては十分すぎる分解能をもつ一方で, 低音部での分解能が悪いという特徴を持っている. 人間の聴覚が対数的なスケールをもっているという事実を考えると特に低音部における周波数の分解能を上げることが求められる. その対策として, 複素スペクトルの位相成分に注目した複素スペクトルの内挿法 [原 83] や各周波数における分解能が等しくなるように窓関数に工夫を加えた Wavelet Transform [KM88] などの方法が採譜処理システムに用いれてきた. そのほか, 非定常信号の分析については, クロス項と言う問題はあるものの Wigner 分布による手法も注目されている. また最近, 近接周波数の分離法という有望な手法が高沢らによって報告されている.

採譜処理は入力した信号の周波数分析を行なった結果を時間方向にならべた時間周波数マップを解析していくという作業であるととらえればわかりやすい. ピーク周波数を表現した時間周波数マップ (複素スペクトルの内挿法によりえられたもの) を以下に示す (図 3.10) [片寄 90a][片寄 88]. 時間周波数マップを解析することにより, 基本周波数成分を正規化された時間に割り当てていくことが採譜の課題であると考えることができる.

周波数空間における楽音への群化問題の最も簡単なレベルは単一楽器の単旋律を対象にする場合である. 通常の楽音においては基本周波数に最もパワーが集中するので, 平均振幅差関数や GSD でのピークを検出すれば目的は達成される. ハミング入力で歌唱の入力を行なうもの [F+89] や音高情報の MIDI 出力を出すというもの (ローランドはなうたくん) など商品化に至っているものもある.

実際には上記のような楽音においてもそれぞれの音高において倍音構成比率が異なる. 低音部においては基本におけるパワーが倍音成分に比べて小さいということがあるため, 思ったような結果が得られないことが多い. また, 人間の声を対象にした場合には, ビブラート成分や音の揺らぎといった成分を考慮にいれないと採譜は出来ないという問題もある. 新原らは, 音の安定部分での平均周波数を用いることでピッチを判定している [新原 84] (図 3.11). さらに, 童謡という

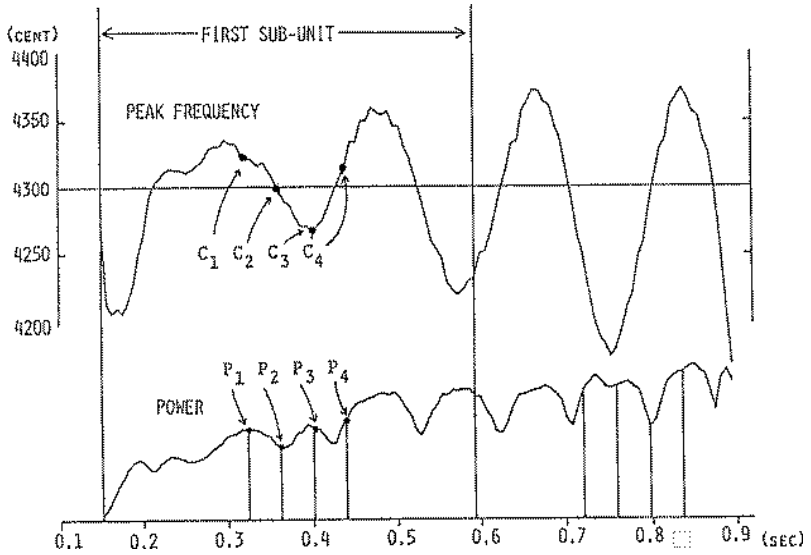


図 3.11: ビブラートに対応した音高の同定 [新原 84]

条件を加え, 調性からはずれた音高の再割当を行なうということで認識率を上げている.

次のレベルとして考えられるのは単一楽器のポリフォニーである [Moo75]. ポリフォニーを対象とした場合, 倍音成分に一致したものがある場合その成分におけるパワーに関して不確からしさが生じる (位相によって変化しうる). 実際の演奏における楽音については微妙な周波数・時間のずれが存在していると考えられる. これらのずれと倍音構成比率がポリフォニーを解析するための特徴量と考えることができる. 現状では信号処理の制約から微妙な周波数・時間のずれを抽出することは難しく, 倍音構成比率に着目したアプローチが多い.

Moorer は平均振幅差関数を開発した周波数解析を行なった後, 2 声のギター音という制約をつけて採譜を行なうシステムを開発している. そのほか, ポリフォニー音楽の採譜については CCRMA で積極的に取り組まれている [CJK+85].

Chafe らのシステムでは, Bounded-Q 周波数変換という手法を用いて抽出した時間周波数マップから低音の部分に現われる音から順に基本周波数と仮定してゆき, 倍音成分に当たる成分を除いていくという処理基本周波数を抽出している.

片寄らはそれぞれの音高での倍音成分のモデル・エンベロープモデルを用いることにより, 同時になる音数に制約を課さない採譜システムを目指している [KI90c]. ここでは, パワー値の微分値を用いで時間方向でのセグメンテーションを行ない, 次に基本周波数の同定を行なっている. 基本周波数判別の閾値は, ある周波数成分に注目したとき, それが基本周波数成分と仮定したときの倍音成分 (高音部) と, その周波数を倍音成分としてもつ基本周波数 (低音部) の大きさからもとめており, ピアノ音に関する認識率はおよそ 90% である. ピアノのほか, ギター, 三味線の音のモデルが用意されている.

Tanguiane は各周波数成分の (パワーではなく) オン・オフが抽出された時に基本波を抽出するというタスクにおいて, コードの推移という条件を使った方法を提案している [Tan88] (図 3.12). ある和音が鳴っているときの基本波および高調波のパターンを s(i) とする. ここで, i は半音階を単位とし, 音階 i においてパワーが存在するときに s(i) = 1, 存在しないときに s(i) =0 となる関数である. 連続するコードにおけるパターン s1(i) と s2(i) の相関関数を計算し, 変化した周波数構造パターンに注目することにより基本波を抽出する. この方法は変化に注目するという点においてロバストである半面, 倍音成分のオン・オフを決定する閾値をどう決めるかという問題があ

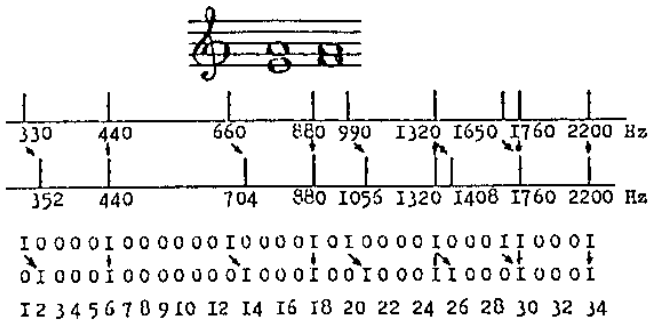


Fig.6.

Table 1.The most salient patterns of partloading intervals between chords (eI,aI) and (fI,aI) for harmonic voices with 5 partials.

Salience of the pattern (correlation)	Size of the pattern in semitones	Numbers of frequency bands, which contain partials, generating the pattern of interval	Number of successive harmonics in low-level patterns	The pattern of interval in standard notation
6	-4	6 18 25 29 30 34	5	(aI,fI)
5	0	6 18 25 30 34	5	(aI,aI)
5	1	1 13 20 25 29	5	(eI,fI)
5	5	1 13 20 25 29	5	(eI,aI)
2	-7	13 25	2	(e2,aI)

Table 2. The most salient patterns of harmonic intervals in the chord (eI,aI) for harmonic voices with 5 partials.

Salience of the pattern (correlation)	Size of the pattern in semitones	Numbers of frequency bands, which contain partials, generating the pattern of interval	Number of successive harmonics in low-level patterns	The pattern of interval in standard notation
9	0	1 6 13 18 20 25 29 30 34	5	(eI,eI)
5	5	1 13 20 25 29	5	(eI,aI)
3	7	6 13 18	2	(aI,e2)

図 3.12: 周波数構造変化パターンから基本周波数を求める過程: 文献 [Tan88] より

る.

McGee は対数的セットした 85 個のフィルターバンクを用いたリアルタイムのポリフォニーの採譜シシテムを発表している [McG89]. ノートの検出という問題に関しては, 高調波成分は重要は思ったほど重要でないという立場から簡単な閾値処理で求めている. また, 同時に鳴った複数音に対するコード名の割当, キーの抽出が行なわれている.

楽音の種類 (数) および音数という条件において, 最も難しいのが, 音色の異なる複数の楽器を対象とする場合である. この条件においては, 基本周波数の同定という作業の他に楽器の推定という作業が加わる. 対象となる楽器種が限定されていても, 音数に制限のない場合は, 解けない問題である. 一般に重積した信号を分離するという問題を音源分離という. 特に, マルチマイクを用い, ノイズ成分を除去するという目的をもった音源分離の研究が早くから進められてきた. Mitchell は 4 個程度のマイクロホンを用い, 音源と雑音源の位置を既知としてマイクロホンを配置して、加減算や絶対値演算によって雑音をキャンセルし、目的の音源からの音を強調する方法を用い雑音成分の抑制する手法を提案している [MRY71]. この方法では, 音源の位置が既知というかなり厳しい条件が入っている. これに対し Flanagan らは, 鋭い指向性を持つマイクロホンアレイを用意し、指向性の頂点の方向を変えながら空間内を走査してことにより音源位置の推定を行なった [FJZE85]。Flanagan らの方法では信号成分がノイズと同じ方向から来ている場合にうまく分離できないという問題があった. 永田らは多数のマイクロホンを広く分布させ、それらの焦点を一点に結ばせる方法に音源位置およびそのスペクトルを推定することができるを行なっている [永田 91]。

柳田らは、音源の数が既知という条件下で, 音源数以上のマイクを用意し, 線型方程式として解を求める手法を提案している [柳田 81]。寺尾らは制御理論におけるシステムパラメータの推定という観点から音源分離を行なう研究を行なっている [寺尾 91]. ここでは, 信号の重積現象を $y(t) = HS(t) + n(t)$ ($y(t)$ は観測される観測出力ベクトル, H は変換パラメータ, $S(t)$ は入力信号 (音源) ベクトル, $n(t)$ はノイズ (但し, 音源数 $\leq$ 観測数)) と置き, $y(t)$ のみから H を同定するというプロセスにより, $S(t)$ の計算すなわち音源の分離を行なっている.

音源分離関係の研究の多くはそれぞれの音源が複数のスピーカーから出力され, すなわち, 音源の数≦マイクの数といういう前提で実現されてきた処理である. その一方で人間は, モノラルスピーカーから出力される複数の楽器音を判別する能力をもっている. これは, 人間がもっている楽音の各種のモデル (パラメータ) やその他の音楽的知識を利用していることにほかならない. ヘルムホルツ以来, 楽音を特徴付ける最も重要な要素は音色であり, 音色とは倍音成分の構成比で決定されると言われているが. 実際には, そのほかにもエンベロープやビブラートなどのその楽器の物理的形状から指定される奏法そのものが楽音を決定する重要な要素となっている. 例えば, エンベロープの場合, ピアノ音において立ち上がり部分をマスクしてやるとそれをピアノと認識するのはかなり困難である. 片寄らはハーモニックエンベロープ (それぞれの倍音のエンベロープ) のモデル, ビブラートの有無を利用したピアノ (複音), クラリネット, バイオリンの採譜システムの基礎実験を行なっている [片寄 91]. また, 対象となる 2 種の楽器がシステムに与えられた時には, システムが自動的に二つの楽音のモデルを比較し, より顕著な差を持った特徴量に優先度が与えられるという処理が実現されている.

柏野らは, 人間が楽器音のモデルを学習しているという観点からボトムアップ処理に重点をおいた研究を進めている. 確率という見地から倍音成分のミスチューニングやオンセットタイミングのずれに関する人間の知覚特性を測定し、それらを実装するというアプローチで音源分離システムを開発している [KT91].

最近では, ニューラルネットを用いた音源分離に関する研究も盛んになってきている. 阿部 [阿部 91] は 4 層の, ニューラルネットをもちいて 2 種類の楽音の混声音からの各楽器音の分離・同定を目指している. これに対し, 秋田 [秋田 91] らは前処理としてクラスター分割を行ない, 入力情報を圧縮することで学習時間・ネットワークのノード数を減らすことを目指した研究を行なっている. これら, ニューラルネットワークを用いた方法は楽音を特徴付ける要素のうち周波数構造に着目したものであるが, 今後, ハーモニックエンベーロープに対して適用することにより, より認識率の高い音源分離システムへの発展が期待される. 音声情報処理において一般に利用されている隠れマルコフモデル (HMM), 時間遅れニューラルネットワーク (TDNN)[Wo88], 学習ベクトル量子化法 (LVQ) によるネットワークパラメータの学習などが参考になろう. 音源への群化におけるもう一つの問題に採譜を行なう時の各種パラメータの設定問題がある. 上記, ニューラルネットワークはパラメータの設定という問題には最も有効な手段の一つである. そのほかに, 周波数構造のパラメータのモデル化という問題に関し, 倉富らはポリフォニーの楽曲を対象にまず基本波となるような成分を抽出し, その基本波に対して, 他の音の重ならない倍音成分の比率を順次見つけてゆくという手法を用い, 1 曲を通して, 教師なしで音の周波数構造を求める手法を提案している [倉富 93]

実用化という観点からは人手で各種パラメータを調整するという手段が取られよう. 今までの採譜研究は主に大学研究機関において行なわれてきたが, 楽音のモデルに関しては楽器メーカーのノウハウがはるかに大きい. これらのノウハウは楽器メーカ資産の中核となるものであろうが, 今後は自動採譜の技術を高めていくためには楽器メーカーとの共同作業が望まれるところである.

3.2.2 音楽分析

上記のサブセクションでは, 楽音への群化, すなわち, 楽音の認識という部分に関して書いてきた. もちろん採譜という処理においては, 楽音がそれぞれに分化した形で抽出される必要があるが, それだけでは単なる音の羅列である. 採譜処理を完結するためには, リズム (拍・および拍子) や認識, 調性の認識を行なう必要がある. このような分析は楽譜の表記のために必要と言うだけではなく, 信号処理部において不十分な部分を補償するためのという処理としても重要である.

音楽分析は音楽学や音楽心理学でかなりの研究が行なわれている. 本来, このような分析は以上のような領域や認知科学に踏み込んだサーベイを行なわなければならないが, 本稿では, 採譜システムを作るという前提で直接的に触れたれたものを中心に, 以下, リズムの認識, コードおよび調性の認識について述べていく.

リズムの認識

リズムの認識は大きく分けて, 音の長さ (音長) の量子化問題と拍子の認識に大別することが出きる. 人間の演奏には意識的なアゴーギク (テンポやリズムにおける微妙な表現) やゆらぎが入る. 前者はそのような長さにばらつきのある音の並びを簡単な整数比にわりつける処理である. 一方で後者は基本となる拍の長さとまとまりの単位を求める処理と考えることが出きる.

量子化問題については自動伴奏という観点からはビートトラッキングとして取り組まれている. 最も単純な実現法はノーテーションソフトウエアにみられるように固定の閾値で振り分けていく方法である. この方法は簡便ではあるが, 情緒豊かな演奏においては誤認識が大きく, 以下に示すような研究が提案されてきた.

Longuet-Higgins[RH87] はいわゆるロールピアノのロールに書かれている単旋律の音高, 音の始点終点の情報を入力として, 音価, 調性の解析を行なうプログラムを作っている. 音楽のリズムは構成上, 2 分木または 3 分木により全体の音価を分解していく過程でできあがるものであるという前提とテヌート・スタカートというような情報を用いて音符の階層的記述を導出している. 基本的には固定的な閾値処理であるが, 2 分木であるか, 3 分木であるかという強い制約を用いておりかなりの認識率が得られている.

Mont-Reynaud[DMR87] は主として MIDI 楽器における自動伴奏という観点からビートトラッキングを実現している. ここでは, (時間により変化する) テンポ情報を用い, 予想されるダウンビートで 1(-1), その半分で 0 となるような直線的な関数 (のこぎり波) を作り, その絶対値でビート位置の確信度を計算している. 尚, 前の音列情報から近いものに対しては大きな重みをもたすような形 (指数関数的なもの) を用いて, テンポの変動に対処している.

音響信号からの採譜といういう例では, ノイズ成分・ピーク周波数検出ミスなどにより音の立ち上がりが誤抽出されるという問題に対処しなければならない. 片寄 [片寄 90a][片寄 88] は上記の方法と基本的には同じであるが, あいまいな音の立ち上がり候補 (パワー成分を微分したものの内小さいもの) に対しては, 先読みを行ない, 基本拍の整数倍付近で明白な音の立ち上がりの検出を行なった上で再度解析を行なうという手法を用いている. 閾値については予め大きく設定しておき徐々に小さくしていくという Course to fine 的な手法をとっている. 同時にフレーズ解析も行い, フレーズの終わりでの長音は長めに演奏されるというヒューリスティックルールを用い, 認識率を高めている (図 3.13).

Desain[DH89] らは弛緩法をコネクショニストモデルアプローチで実現するというユニークな方法で量子化を実現している. ここでは, 隣接する各音の間隔の長さの比が有理数に近づかせるようにする変化関数を作り, その変化関数の働きをネットワーク上の相互作用として実現し, イタレーションを行うことにより最終的に拍の量子化を行っている. また, 個々の音だけではなく, 連続する二つの音の長さの和を一つのノードとしてネットワークの相互作用に組み込むことで, 処

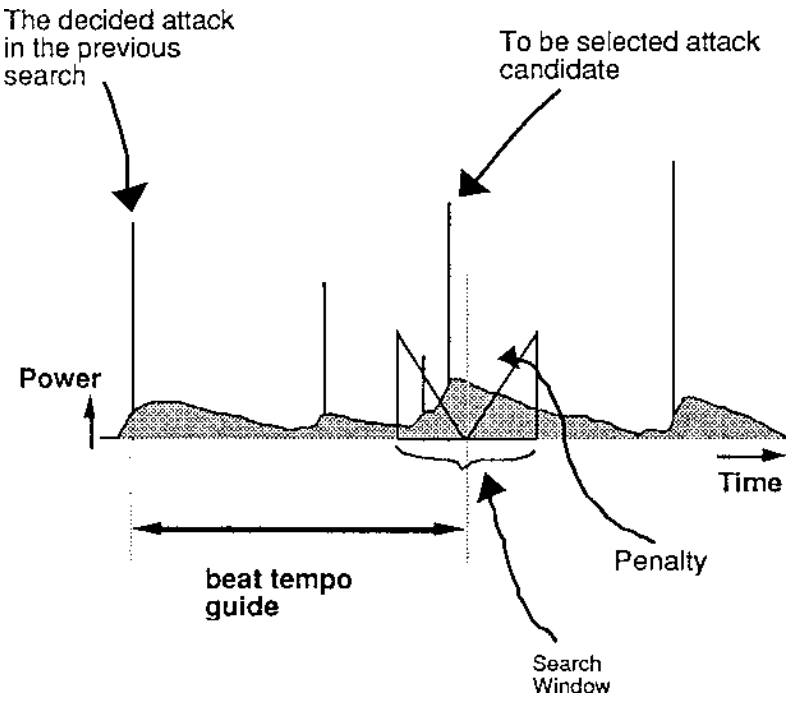


図 3.13: 次拍の追跡

理の向上が図っている．この方法はリアルタイム処理には向かないがかなりロバストな方法である (図 3.14, 図 3.15).

以上, 音符の量子化, ビートトラッキングという処理について述べてきたが, もう一つ重要な課題が拍子の割当である．拍子の認識という問題は単純化すると, 2 拍子系か, 3 拍子系かという問題になるが, 例えば 3 拍子か 6 拍子かという問題は音楽を経験者でもしばしば間違うことがある高次の音楽認知問題である．採譜問題において取られてきたのは筆者の知るかぎり単純化した方法である．簡単な音楽 (簡単というのには若干の問題もある) においては一般的に 1 拍目にアクセントが与えられる．ここでのアクセントとは音が大きいとか, 1 拍目でたくさんの音がならされるとかベース音が与えられるということを表す．このような条件が成り立つような楽曲の採譜においては, 装飾音を除く最小長さの単位とする (音の大きさ, 数, ベース音候補) の配列を作り, ビート n(例えば, 2, 3, 4, 6, 8) でわった剰余が一致するインデックス群の配列要素の総和に n をかけ, 最もおおきいインデックス群の n から拍子を求めることができる [片寄 90a][片寄 88]．このときに剰余が 0 でなければ弱起の楽曲であると判断できる．このような方法は, 一般的な音楽というところまで範囲を広げると明らかに限界がある.

採譜というテーマからは離れてしまうので詳しくは述べないが, 音楽認知科学の範疇では, Lerdahl [LJ83] がシェンカー流音楽理論とチョムスキー流の文法を概念的に結合した音楽の構造解析法が示している.

聞き手の認知という観点から Meyer[Mey56] は「一つのリズム・グループが一つのアクセントしかもたない」という原則と音のまとまり方に関する心理学的知覚ルールを用いた音楽解析法を示ししている．この方法の長所は音楽の構造理解そのものに認知的要素を考慮したことである．が, 計算理論という観点からはこの長所自身が欠点となっている．すなわち, この方法の根拠とするアクセントの決定論自身に音楽センスともいうべき主観的要素が要求されるために, 結局のところ完全な数理的構造解析法を示したというまでには至っていない.

また, Meyer は当初小節という単位をアクセントの解析ひいては構造解析のよりどころとして

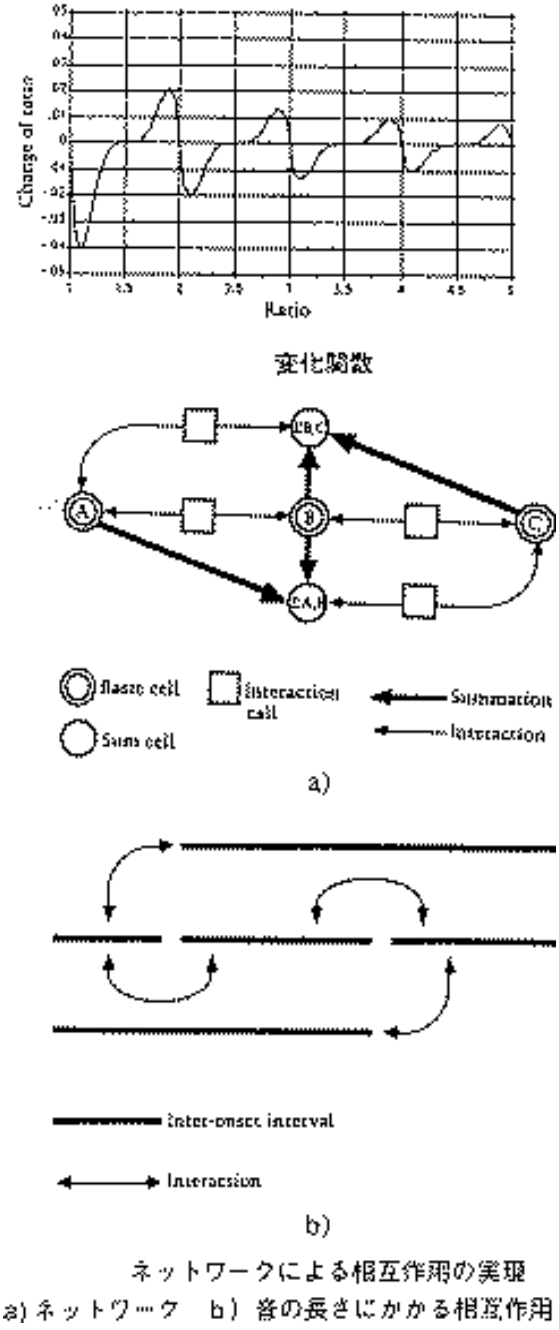


図 3.14: 弛緩法による音長の量子化処理, [DH89] より

いたという問題もあって, 後に弟子の Narmour とともに上記の構造解析法にかわり, 音楽進行に於ける局所的な期待感のボトムアップ的な重ね合わせが音楽構造を作るという構造解析法を発展させている [Nar77].

村尾 [村尾 92] はナーモアの方法が複雑な記号付けを行なわないと構造の解析が出来ないということといくつかの内在する矛盾に解決する手法として, 音長, 音形, 類型などの構造からストラクチャーを解析する手法を提案している.

一般に音楽の構造と拍子は密な関係を持っており, フレーズがわかれば (フレーズの長さが重要なキーになる) 拍子が決定できるといわれている. が, 上記の構造解析の方法は暗黙のうちに楽譜を前提に理論が進められていたり, また, 基準となる拍を前提にしているため, 拍子を決定する

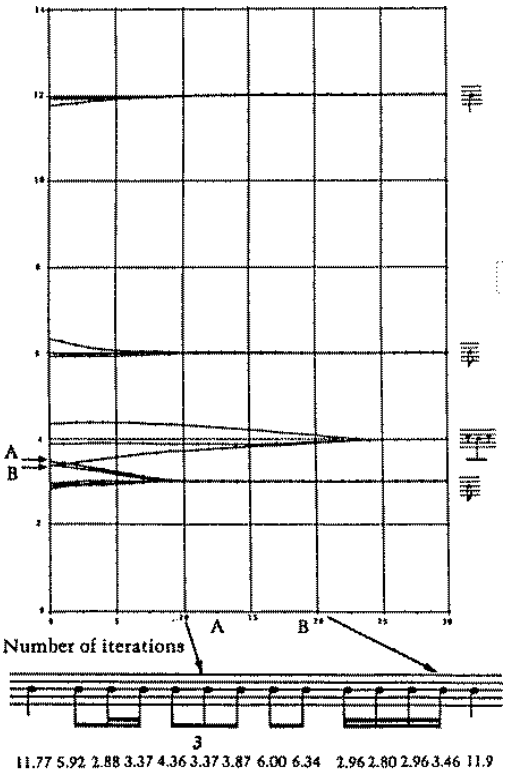


図 3.15: 弛緩法による音長の量子化結果, [DH89] より

完全なコンピューテーショナルモデルは存在していないという状態である．ここから先は私見になるが, 人間が自然な長さと感じる拍の長さとその拍の長さによって解析されるフレーズの長さがそれぞれにエネルギー場を持っており, それぞれの相互作用として定義されるエネルギー場で最も安定となる位置において拍子が決定されるのではないかと考えている．

コードおよび調性の認識

採譜という処理においては, 調性の判別は主としてマッチング手法により実現されたものが多い．長調階 (およびその 3 度下の自然的短音階) については, 半音の現われる位置 (各スケールに 2 個) は一定しており, その部分が検出できれば調性が判別される．また, (5 度圏) 近親調においては半音の位置が一つだけ変化するという条件を用いる方法もある．得られた音列と 24 種の音階 (主音 (12)* 長短調 (2)) とのマッチングを行い, 主調とその近親調で, 大きな値をもつものを選んでやることで, 主調を決定することができる．これらの方法は主に単旋律を対象に用いられる方法であるが, コードを積極的に利用する方法もある．例えば, ドミナントモーションが発見されれば調性を決定することができる．また, 旋律的短音階や和声的短音階の場合上記のヒストグラム法では認識が難しいが, コードの推移性という条件を用いることで比較的容易に調性を決定することができる．もっともスケールは各民族や分化圏において固有のものがあるので, 対象に応じたマッチングを実現すべきである．

コードの認識に関しても, 採譜という研究においては, 基本的にはマッチング処理で実現されたものが多い. 片寄は 3 音またはキーが決まっているときには 2 音でのマッチングでコードの認識を行なっている. アルペジオなどのために音がばらけるときには, 上昇パターンにおける最も低い音をベース音と仮定して後続の音を待つ. この操作ではコードが細かく分割しすぎることがあるため, 読み変え, 併合などのルールにより妥当なコードの数まで減らすという処理を実現している.

コンピュータによるコード認識については, 採譜という観点より, むしろ, 音楽のアナリシスの支援として取り組まれたものが多い. その最初のものはおそらく Winograd によるバッハのコラールの解析である [Win68]. Winograd は branch-and-bound, left-to-right による手法でコードの解析を行なっている. 70 年代に後半には, Schenker の音楽理論を用いたコード解析が精力的に取り組まれた [Smo80]. その後明示的なルールベースによるバッハコラールの解析やジャズの解析などが各地で行なわれてきた. コンピュータを用いた音楽アナリシスの研究は Understanding Music with AI[MBL92] という本の Analysis という節に紹介されているので詳しく知りたい人はそちらを参照されたい.

上記のコード, 調性の認識を初め, メロディの抽出といった問題は人間の深い認知活動に関係し, 音楽学, 心理学, 認知科学など現在まさに進行中の研究領域である. ジャンルを越えて有効な音楽認識のコンピューテーショナルモデルは現段階では存在していない. 特にシステムとして実現されたものは, 人間の音楽的活動のほんの一部 (表層) しか扱えていないということを一言述べておく.

3.2.3 楽譜化

上記のような分析を行なうことによって入力された音列データは基本的には楽譜に近い形となっている. このようなデータから音分の割当, シンコペーションの表現, 和音, 連符のマージング, 休符等の挿入等をおこなって楽譜を出力するためのシンボルデータが生成される [片寄 90a] [片寄 88]. 作成されたシンボルデータに対し形状を定義し, 位置決めを行なうことによって楽譜データ (グラフィックス) ができるわけである. このような作業をもとにピアノ音を採譜した結果をしたに示す. なお, 実際の音響においては残響等が存在する. この楽譜はコードを解析した上で, 和声学的にみておかしいと思われる音の長さを補正したものとなっている (図 3.16) [片寄 90a][片寄 88].

楽譜出力ということに話を絞ると, 最近ではノーテーション専用のパソコンソフトウエアが発売されている. ほとんどのノーテーションソフトウエアにおいては専用のデータフォーマットのほかスタンダードミディファイルなど事実上標準化されたデータプロトコルの入力が可能である. 採譜した結果をそれらの形式に変換できれば簡単に楽譜化が出きる. 譜割, 位置決め等に関しては人間の専門家に及ばないという問題もあるが, 人間のオペレーションを前提として調整機能も拡充されており, 現段階ではほぼ完成した領域であると言うことができる (図 3.17).

3.3 演奏表現の生成 (片寄)

音楽情報処理関係の研究の中でも最も積極的に取り組まれてきたのが自動演奏である. 一口に自動演奏と言ってもさまざまなタイプのものがあるが, ここでは, クラッシックに代表される西洋音楽を題材に情緒溢れる演奏を行なうという視点からの研究をサーベイを行なうことにする. これらの研究は音楽解釈という問題に深く関係するものである. また, 局所的な音列データがどのように表現と言ったテーマは音楽心理学の分野で詳しく取り扱われているが, ここでは ICMC で発表されたものを中心に紹介する.

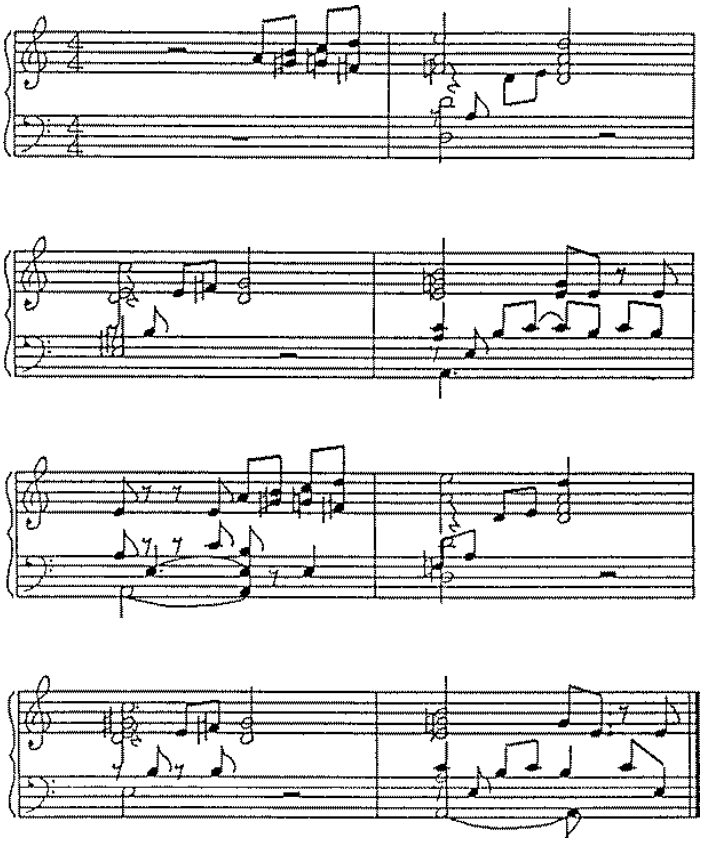


図 3.16: ピアノ楽譜採譜例

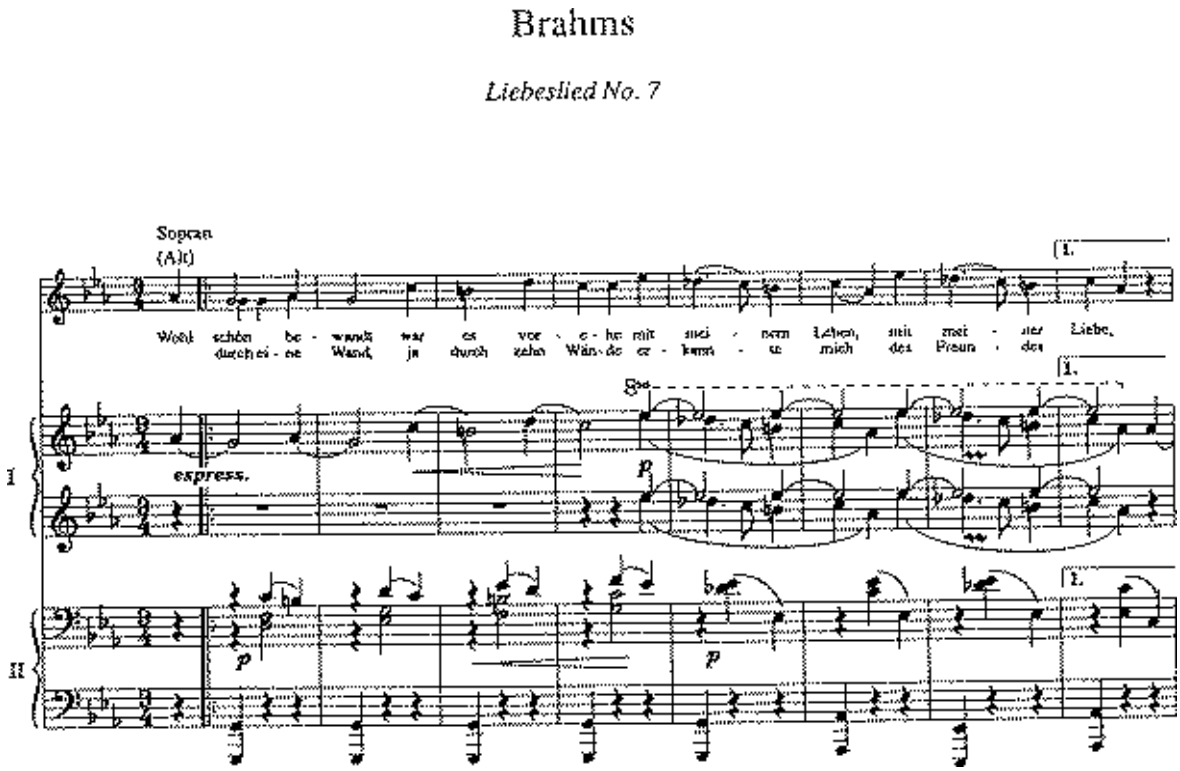


図 3.17: ノーテーションソフト Finale による楽譜出力例

情緒あふれる演奏データの生成という研究に関して, 計算機を用いた初期のものは 1984 年に発表された Sundberg ら, Clynes の研究である.

Sundberg, Friberg ら [FS84a] は, コード, 調性などと音の大きさ, フレーズ間の間の関係といった演奏ルールをアナリシスバイシンセシンスの方法で作成するという研究を行なっている. 演奏ルールについてはさらに詳しい報告が 1991 年のコンピュータミュージックジャーナルに記載されているので興味のある人はそちらを参照されたい [Fri91].

Clynes[Cly84a] は, 階層的な表現が楽曲の演奏を決定するという考えの元に立った解釈システムを提案している. 例えば, 3/4 拍子の楽曲のある 4 小節が 48 個の 16 分音符から成り立っている場合,

- それぞれ 4 つの小節に対する音量
- 支配時間の割当, 1 小節内での各, 1 拍目, 2 拍目, 3 拍目の大きさ
- 支配時間の割当, 1 拍内のそれぞれの 4 つの 16 分音符の音量
- 支配時間

を比で表現し, その積算をもって演奏データを作成している. その他に, メロディに対する表現, ビブラートのかけ方に対する演奏ルールが議論されている.

Wessel[WBS87] らは, 下記 Todd の文献をリファーしながらフレージングとアーテイキュレーションのコントロールをシステム的実現という観点から取り上げている.

Todd は 1985 年に Lehrdahl, Jachendoff の time-span reduction に基づいて, フレーズの最後を延ばすという効果のモデリングを行なっている [Tod85]. さらに, 階層的なフレーズ構造に対し, 放物線のフィッティングを行ない, その効果を実際の演奏と比較するためのシステムを構築している [Tod90].

演奏生成というアプローチからは若干異なるが, Desain は演奏における相対的な音符の長さ (情緒ある演奏においては変化する, 表記上のものより長く弾かれたら大きな値となる) の 1 次元波形に対し, 自己相関を取るツールを作成し, 演奏から拍構造を検出する手法を提案している [DdV90].

Bresin らは今まで報告された研究の多くがシンボル的な処理であるのに対し, ニューラルネットワーク的なアプローチでの演奏生成システムについて検討を行なっている [BPV92]. ここでは, 上記 Sundberg らの演奏ルールの実現などについて述べている. そのほか, 1992 年の ICMC においては, 表情付けに関するツールが 2, 3 発表されている [Hon92], [Opp92], [Ruf92]. 興味のある方は参照されたい.

一方, 国内では, 甲南大学 (田口ら), 大阪大学 (片寄ら), 筑波大学 (富塚ら), などがこの分野に相当する研究を行なっている. 田口は MUSE というピアノ用階層的演奏記述言語を開発し, ワルツの伴奏表現の数理的な解析を行なっている [TO89]. さらに, 最近では主観的な評価と演奏表現の関係に関する検討を進めている [TO92]. 富塚らは, 予め, さまざまなフレーズの演奏法をデータベースとして登録しておき, 自動演奏を行おうとする楽曲の同等あるいは類似のフレーズに対応する演奏法をデータベースから検索し, つなぎ合わせるという手法の解釈システムを作成している [富塚 88].

MIS は基本的には上記の Clynes, Todd の同様の機構をもつ演奏生成システムであるが, 演奏ルール抽出過程をもっている [KI90a]. 類型は同様に演奏されるという仮定から, メロディから多重の類型候補を作っておき, 演奏の特徴 [高見 89] に一致するものがあればその演奏法を抽出を行なうという手続きが取られている. なお最近では, Todd に代表されるフレージングルールの限界に対し, アクセント代償理論 [村尾 92] の導入検討を行なっている [片寄 93].

3.4 感性情報の処理 (片寄)

従来の情報処理において論理的・客観的知識情報が中心におかれていたことに対し, 情緒的な情報の処理を目指した領域として感性情報処理という研究領域が確立しつつある. 関連領域としては認知科学があるが, 認知科学が人間の知能や感性といったもののの理解を目指しているのに対し, 感性情報処理においては, 感性や情緒といった情報を担うデータの計算機上でのハンドリングを中心課題としており, より現実性という点に注目した研究領域であるということが出きる. 具体的には,

1. 形容詞的情報のハンドリング
2. 人間が実際に接している形での情報の入出力
3. インタラクティブ性 (リアルタイム性, 自立性)

の機械的実現をめざしたものが感性情報処理の範疇に入る研究 (システム) と考えることが出来よう. 2, 3 については仮想現実感の尺度とされる Presence, Interaction, Autonomy[Zel91] に置き換えられるものである. これらはそのシステムを使う人間の感性に刺激を与えるという点で重要な要素と考えられる. 音楽は芸術という領域に属するものであり, 感性情報処理の対象としていち早く取り上げられた分野の一つである. 音楽における感性情報処理システムの形態を, 上記の分類に従って整理する.

1. 形容詞的情報のハンドリングを中心に扱ったもの
   (a) 演奏からの形容詞情報の抽出システム
      - ピアノ演奏 (ポピュラー) 音響から, コード進行, テンポなどを解析し, 形容詞情報を抽出したシステム [片寄 88]
      - ジャズ (MIDI 信号) からリアルタイムでコード, キーおよび局所的テンション値 (エモーションファクタ) の抽出を目指したシステム [金森 92]
   (b) 形容詞情報を利用した作・編曲 (支援) システム
      - 感性パラメータ (因子分析によって求められた形容詞座標空間) を用いた作編曲支援システム [SKI89a]
      - k-line を用い, エモーション (形容詞情報) をコントロールパラメータとした作曲システム [Rie91]
2. 人間が実際に接している形での情報の入出力システム
   - 採譜や自動演奏など視覚・聴覚・演奏などの機能の統合化を目指したシステム [KI90b]
   - 視覚センサを用いた指揮・舞踏伴奏システム [SKI89b] [1]
   - 解釈 (支援) システム [FS84b, Cly84b, 片寄 90b]
3. ミュージックパートナーシステム
   - ソリスト (人間) に対する計算機伴奏者を目指したシステム[2]
   - 音楽解析部, 演奏部を持ったインターラクティブコンポージング環境 [Row92]
   - ニューラルネットを用いてドラムス, キーボードの掛け合いを目指したもの [NW89]
   - ジャムセッションパートナーを目指したもの [和気 89]

以下, 感性情報処理システムという視点からみて最も興味ある形態の一つと考えられる Rowe のシステムについて紹介を行う.

[1] 加速度センサ等を用いた指揮システムやテンポ入力システムについては後で詳述

[2] これについては 4 章インタラクティブシステムにて詳述 [Dan84][Ver84]

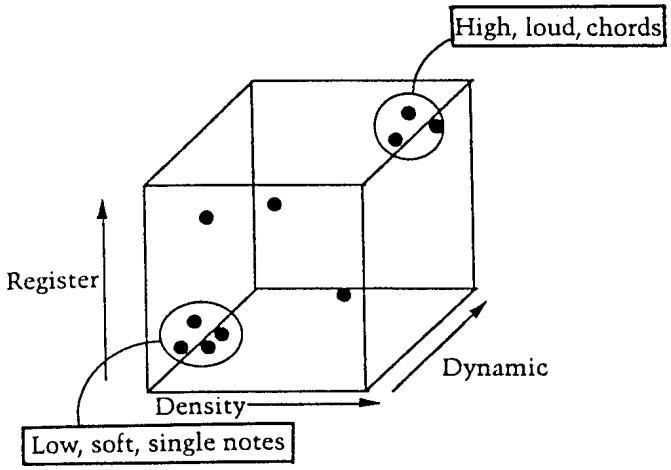


図 3.18: 音域, 強弱, 密度の 3 次元の特徴空間

Cypher 概要

Cypher は Media Lab. Robert Rowe が博士研究として取り組んだ先進的なインタラクティブコンポージング環境である. インタラクティブコンポージング環境としては Puckette の MAX が有名であるが, 以下に述べるように自立的な音楽認識機構が実現されている点で感性情報処理システムと位置づけられるものである.

Cypher はリスナー部とプレーヤー部という要素から成り立つ. リスナー部ではリアルタイムで MIDI イベントを解析して, 音の密度, 音域, 強弱など低次の音楽要素からコード, フレーズなどの高次の音楽要素までの認識を行なう. プレイヤー部においては, リスナー部で認識された音楽要素に対する反応の仕方の記述したルールが発火され, 演奏が生成される. この両者の機能を組み合わせることにより, Cypher はインタラクティブコンポージングツールとして使用されるほか, 究極的にはインプットなしで新しい音楽を作りだしていくこともできる. Cypher の概念的背景には Minsky の Society of Mind がある. この特徴はリスナー部とプレーヤー部の両者にみることができるが, 特にリスナー部で顕著である. リスナー部においては, 一つ一つをとると比較的シンプルな音の密度, 音域, 局所的コードなどを抽出するエージェントが存在し, それぞれのエージェントがネットワークを通じて協調しあうことにより, より高度 (複雑) な音楽知覚・認知を実現するという手法が取られている.

Cypher リスナー部

Cypher のリスナー部は今までの多くの研究と同様に西洋音楽を対象としている. 認識対象としては以下の二つに大きく分類される.

初期認識対象: 6 次元 (密度, 速さ, ラウドネス, 音域, 音価 (デュレーション), ハーモニー (コード)) (図 3.18).
2 次認識対象: フレーズ, フレーズ内における初期認識対象の規則性など (図 3.19).

以下にそれぞれの認識エージェントの概要を示す.

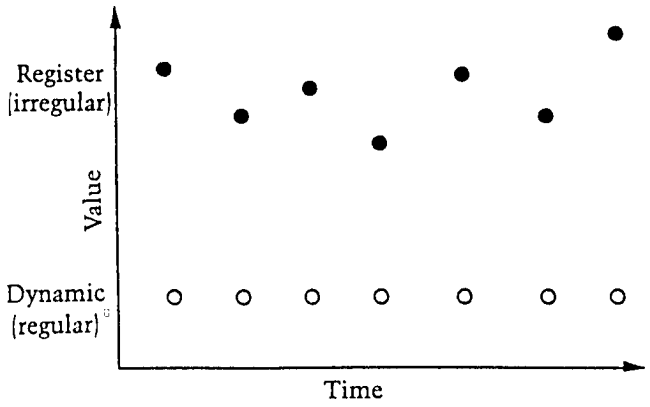


図 3.19: 音域・強弱に関する規則性と不規則性の検出

- 音域 (register)
  浮動音域調整
  音色は考慮しない
  出力としては
  2 オクターブ以内なら　LOW, HIGH
  それ以上なら　LOW, M LOW, M HIGH, HIGH
  コード (複数音が検出された) のときは多数決で決定
- 強弱 (dynamic)
  2 段階 (強・弱) の評価
  コードのときは多数決
- 密度 (density)
  音数 (一つ以上のときはコード検出ルーチンを起動)
  2 オクターブ以内ならナロー
  それ以上ならワイド
- 速度 (speed)
  連続する二つの音のアタック間のタイム
  2.1 秒以上　スロー
  1 秒以上　ミディアムスロー
  0.3 秒以上　ミディアムファスト
  それ以下　ファスト
- ハーモニー (chord) (図 3.20)
  コネクショニストアプローチ (ノートからコードの活性化)
  バッハのコラールに対し 60 密度, 音域, ビートによるチェック
- 調性 (key)
  コード + 密度の重み付けで認識
- ビートトラッキング
  1 拍を 0.28 〜 1.5 秒の間にあるとして認識
  予測した位置に当たっていたら加点
  シンコペーションは少ないというヒューリスティック
  (ノートの長さ又は二つの隣接したノートの長さ)*0.5 or 2 or 1/3 or 1/4

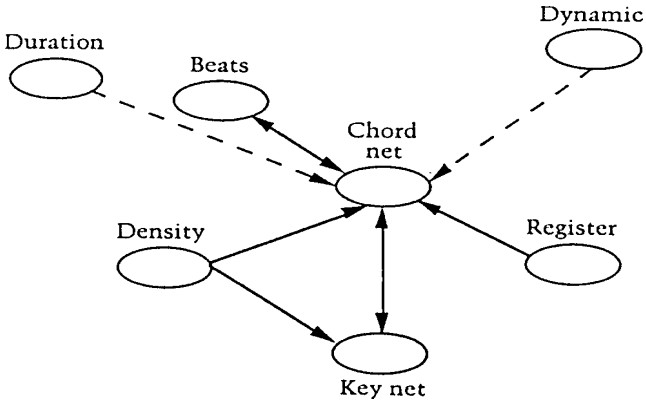


図 3.20: コード認識エージェント

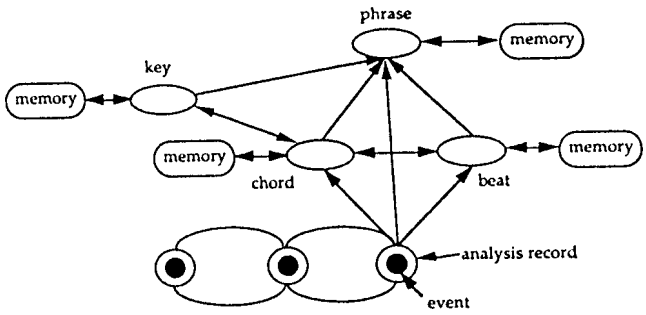


図 3.21: フレーズ, 調性認識エージェント

- フレーズ (図 3.21)
  初期認識対象で認識した結果の変化点で判断
  コードについてはトニック, ドミナントにおいて, 変化点として加点する

図 3.22 の曲を与えて Cypher が調性とフレーズを追跡する様子を図 3.23, 図 3.24 に示す.

Cypher プレーヤー部

ユーザーは図 3.25 に示すようなユーザーインターフェースを用いてリスナー部で得られた認識結果に対する応答の仕方を記述する. Cypher プレーヤー部においてはユーザーが記述した応答の仕方 (認知・実行サイクル) を順次発火していくことにより, インターラクティブなコンポージング (伴奏) が実現される. なお, 認知・実行サイクルについては直列的な結合が可能であり, よりユーザインタフェース性の高いものとなっている (図 3.26).

応答の仕方としてユーザは次の 3 種類を選ぶことができる.

1. シーケンスデータの演奏 (予め入力した演奏パターンの演奏)
   10 秒以内のシーケンス, 1 秒毎に操作可能
2. 作曲プログラムの初期化 (呼び出し), コードの変化, トレモロ等
3. 入力される MIDI 信号の加工
   小モジュールの直列フィルタリング

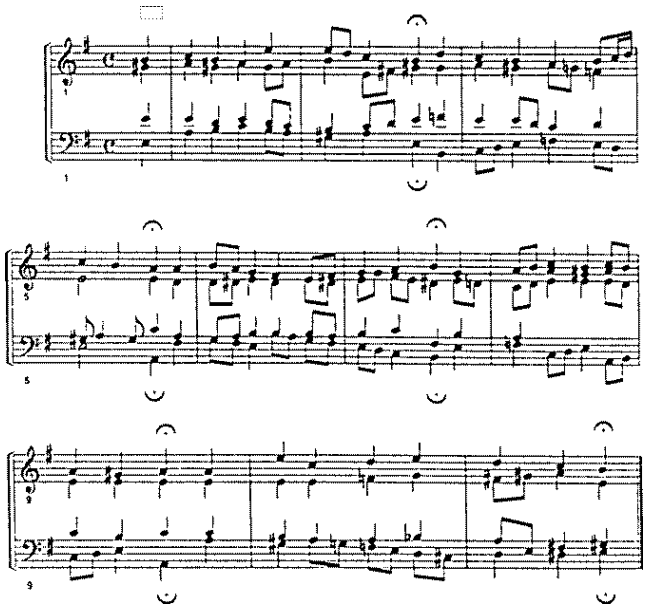


図 3.22: 対象曲 "Ach Gott, vom Himmel sieh' darein", J.S Bach

感性情報処理システムまとめ

深くは述べないが, Cypher ではリスナー部を複数個持つことができ, 自らの出力に対し反応させていくということが出きる. 外部からの入力が来ないときにも“内省的に”作曲可能ということを特徴の一つとしてうたっている. 以上, Cypher を中心に感性情報処理システムを紹介した. 感性情報処理とは確立した領域ではないが, 音楽情報処理の一分野としても最も注目されるものの一つである.

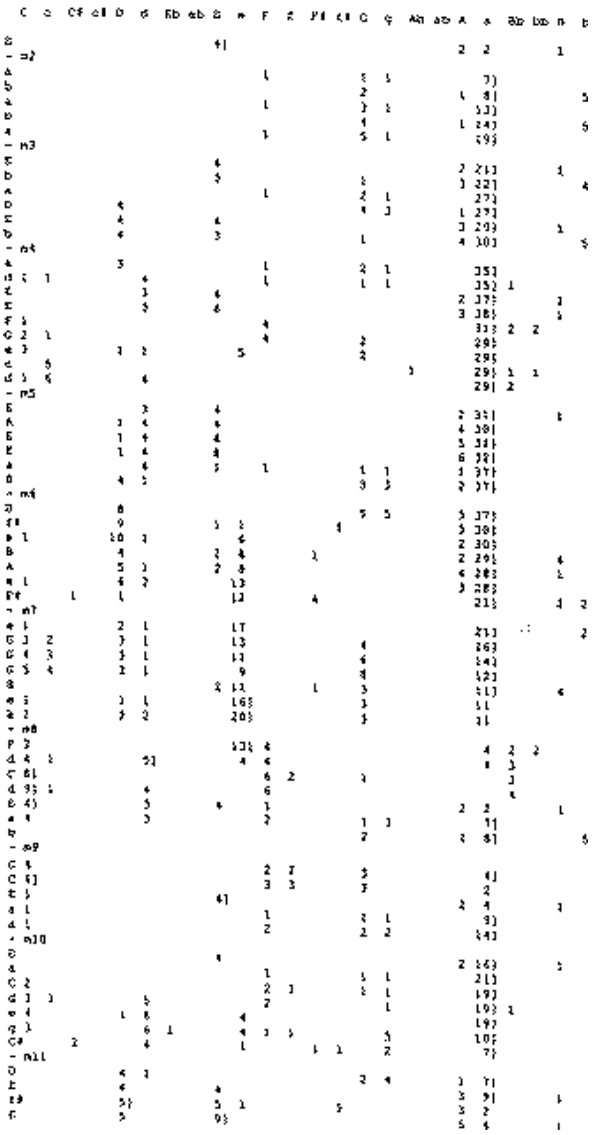


図 3.23: 調性の追跡

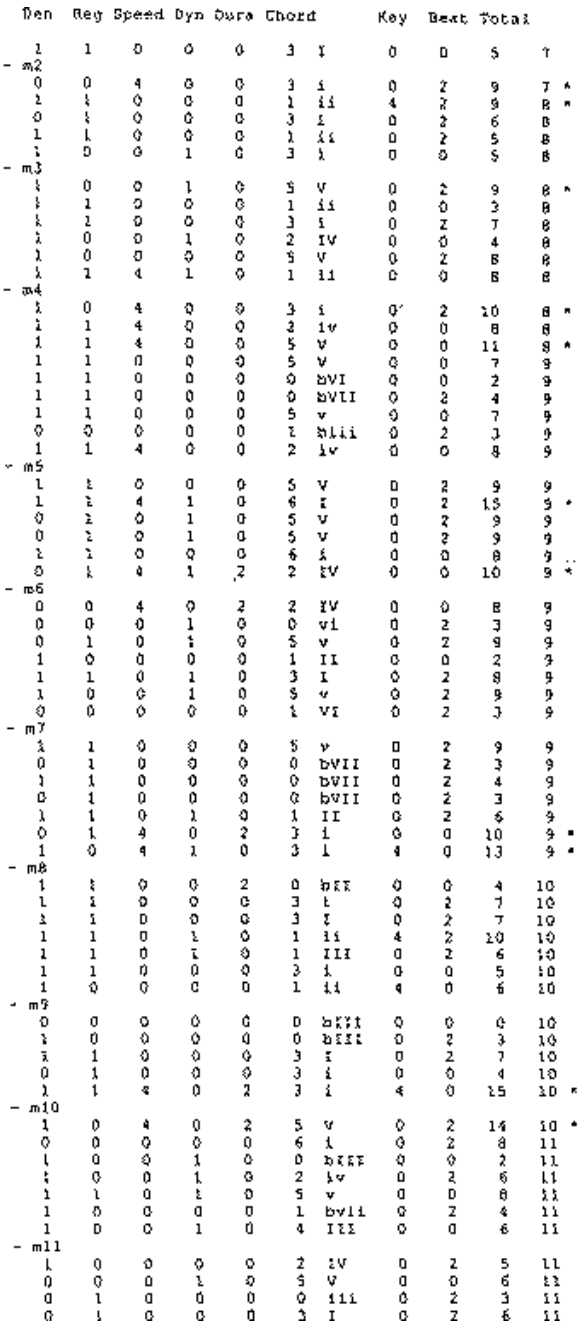


図 3.24: フレーズの追跡

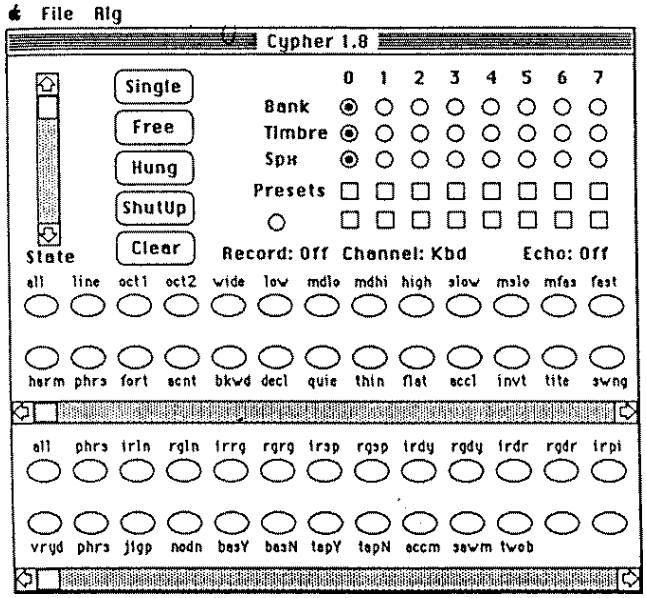


図 3.25: プレーヤ部におけるユーザインターフェース

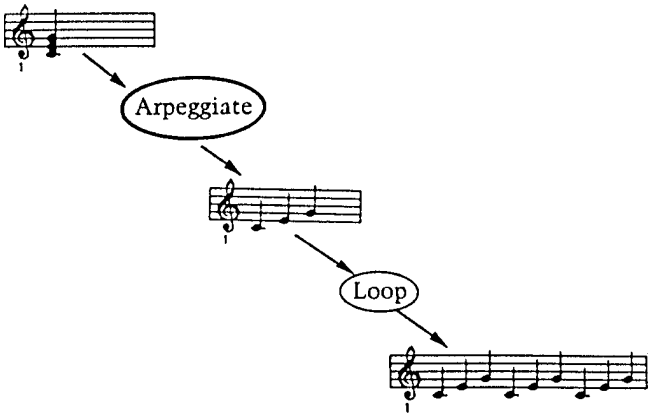


図 3.26: 認知・実行サイクルの直列的な結合

参考文献

[BPV92] R. Bresin, G. De Poli, and A. Vidolin. Symbolic and sub-symbolic rules system for real time score performance. In *Proceedings of 1992 ICMC*, pp. 211–213, 1992.

[CBT92] Alstair Clarke, Malcolm Brown, and Mike Thorne. Problems to be faced by developers of computer based automatic music recognisers. In *Proceedings of 1990 ICMC*, pp. 345–347, 1992.

[CJK+85] C. Chafe, D. Jaffe, Kyle Kashima, B. Mont-Reynaud, and J. Smith. Techniques for note identification in polyphonic music. In *Proceedings of 1985 ICMC*, pp. 399–405, 1985.

[Cly84a] M. Clynes. Secrets of life in music. In *Proceedings of 1984 ICMC*, pp. 225–232, 1984.

[Cly84b] M. Clynes. Secrets of life in music. In *Proceedings of 1984 ICMC*, pp. 225–232, 1984.

[Dan84] Roger Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of 1984 ICMC*, pp. 193–198, 1984.

[DdV90] P. Deasin and S. de Vos. Autocorrelation anfd the study of musical expression. In *Proceedings of 1990 ICMC*, pp. 357–360, 1990.

[DH89] Peter Desain and Henkjan Honing. The quantization of musical time: A connectionist approach. *CMJ*, Vol. 13, No. 3, pp. 56–66, 1989.

[DMR87] R. Dannenberg and B. Mont-Reynaud. Followin an improvisation in real time. In *Proceedings of 1987 ICMC*, pp. 241–248, 1987.

[F+89] H. Fujiiet al. A music arrangement system based on hummed melodies. In *Proceedings of 1989 ICMPC*, pp. 103–106, 1989.

[FAPD92] Ichiro Fujinaga, Bo Alphonce, Bruce Pennycook, and Glendon Diener. Interactive optical music recognition. In *Proceedings of 1992 ICMC*, pp. 117–120, 1992.

[FAPH91] Ichiro Fujinaga, Bo Alphonce, Bruce Pennycook, and Kharim Hogan. Optical music recognition: Progress report. In *Proceedings of 1991 ICMC*, pp. 66–73, 1991.

[FJZE85] J. L. Flanagan, J. D. Johnston, R. Zahn, and G. W. Elko. Computer-steered microphone arrays for sound transduction in large room. *J. Acoust. Soc. Am.*, Vol. 78, No. 5,, 1985.

[FPA89] Ichiro Fujinaga, Bruce Pennycook, and Bo Alphonce. Computer recognition of musical notation. In *Proceedings of First International Conference on Music Perception and Congnition*, pp. 87–90, 1989.

[Fri91] Anders Friberg. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, Vol. 15, No. 2, pp. 56–71, 1991.

[FS84a] L. Frydon and J. Sundberg. Performance rules for melodies: Origin, functions, purposes. In *Proceedings of 1984 ICMC*, pp. 221–224, 1984.

[FS84b] L. Frydon and J. Sundberg. Performance rules for melodies: Origin, functions, purposes. In *Proceedings of 1984 ICMC*, pp. 221–224, 1984.

[Hon92] Henkjan Honing. Expresso, a strong and small editor for expression. In *Proceedings of 1992 ICMC*, pp. 215–221, 1992.

[KI90a] H. Katayose and S. Inokuchi. Expression extraction in virtuoso music performances. In *Proceedings of 1990 ICPR*, pp. 780–784, 1990.

[KI90b] H. Katayose and S. Inokuchi. Kansei music system. *CMJ*, Vol. 13, No. 4, pp. 72–77, 1990.

[KI90c] H. Katayose and S. Inokuchi. Kansei music system'90. In *Proceedings of 1990 ICMC*, 1990.

[KM88] R. Kronland-Martinet. The wavelet transform for analysis, synthesis, and processing of speech and music sounds. *CMJ*, Vol. 23, No. 4, pp. 11–20, 1988.

[KT91] K. Kashino and H. Tanaka. A rule-based approach toward sound source segregation,. In *Proceedings of 1991 ISMA*, pp. 71–74, 1991.

[LJ83] F. Lerdahl and R. Jackendoff. *A generative theory of tonal music.* The MIT Press, 1983.

[MBL92] K. Ebcioğlu M. Balaban and O. Laske. *Understanding Music with AI.* The MIT Press, 1992.

[McG89] McGee. Real-time acoustic analysis of polyohonic music. In *Proceedings of 1989 ICMC*, pp. 199–202, 1989.

[Mey56] L.B. Meyer. *Emotion and Meaning in Music.* University of Chicago Press, 1956.

[MHS+85] Toshiaki Matsushima, Tadanori Harada, Itaru Sonomoto, Katsuhiro Kanamori, Akio Uesugi, Yuji Nimura, Shuji Hashimoto, and Sadamu Ohteru. Automated recognition system for musical score — the vision system of wabot-2 —. In *Bulletin of Science and Engineering Research Laboratory*, pp. 25–52. Waseda University, 1985. No. 112.

[Moo74] J. A. Moorer. The optimum comb method of pitch priod analysis of countinuous digitized speech. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, pp. 330–338, 1974.

[Moo75] J. A. Moorer. On the segmentation and analysis of continuous musical sound. Technical Report STAN-M-3, Stanford Univ., Department of Music, 1975.

[MRY71] O. M. E. Mitchell, C. A. Ross, and G. H. Yates. Signal processing for a cocktail party effect. *J. Acoust. Soc. Am.*, Vol. 50, No. 2,, 1971.

[Nar77] E. Narmour. *Beyond Schenkerism.* University of Chicago Press, 1977.

[NW89] M. Nishijima and K. Watanabe. Interactive music composer based on neural network. In *Proceedings of 1989 ICMC*, pp. 53–56, 1989.

[Opp92] Daniel V. Oppenheim. Compsitional tools for adding expression to music in dmix. In *Proceedings of 1992 ICMC*, pp. 223–226, 1992.

[Pre71] David S. Prerau. Computer pattern recognition of printed music. In *AFIPS Conference Proceedings*, volume 39, pp. 153–162, 1971.

[Pre75] David S. Prerau. DO-RE-MI: A Program that Recognizes Music Notation. *Computer and Humanity*, Vol. 9, pp. 25–29, 1975.

[RH87] H. C. Ronguet-Higgins. *Mental Processes.* The MIT Press, 1987.

[Rie91] R. D. Riecken. Goal formulation with emotional constraints : Musical composition by emotional computation. In *Proceedings of 1989 AAAI (Innovative Applications of Artificial Intelligence)*, 1991.

[Row92] Robert Rowe. Machine listening and composing with cypher. *CMJ*, Vol. 16, No. 1,, 1992.

[Ruf92] Donald Ruffcorn. Livestep: Editing step quantized music from libraries of live performances. In *Proceedings of 1992 ICMC*, pp. 231–234, 1992.

[SKI89a] N. Saiwaki, H. Katayose, and S. Inokuchi. An approach to a computer assisted composition system with kansei parameter. In *Proceedings of 1989 Intl. Workshop in AI and Music*, 1989.

[SKI89b] N. Saiwaki, H. Katayose, and S. Inokuchi. A new approach to music through vision. In *Proceedings of 1989 Intl. Workshop in AI and Music*, pp. 1–10, 1989.

[Smo80] S. W. Smoliar. A Computer Aid for Schenkerian Analysis. *CMJ*, Vol. 4,, 1980.

[Tan88] A. S. Tanguiane. An algorithm for recognition of chords. In *Proceedings of 1988 ICMC*, pp. 199–210, 1988.

[TO89] T. Taguti and M. Ohta. Agogics as a metrical rhythm and phrasing. In *Proceedings of 1989 ICMPC*, pp. 219–224, 1989.

[TO92] T. Taguti and M. Ohta. Computational rendiation of piano music and its subjective evaluation. In *Proceedings of 1992 ISMA*, pp. 203–206, 1992.

[Tod85] Neil Todd. A model of expressive tming in tonal music. *Music Perception*, Vol. ??, No. ??, pp. 33–58, 1985.

[Tod90] Neil Todd. Computational modelling of musical expression. In *Proceedings of 1990 ICMC*, pp. 373–375, 1990.

[Ver84] Barry Vercoe. The synthetic performer in the context of live performance. In *Proceedings of 1984 ICMC*, pp. 199–200, 1984.

[WBS87] D. Wessel, D. Bristow, and Z. Settel. Control of phrasing and articulation in synthesis. In *Proceedings of 1987 ICMC*, pp. 108–1162, 1987.

[Win68] T. Winograd. Linguitics and the computer anlysis of tonal harmony. *J. Music Theory*, Vol. 12, pp. 2–49, 1968.

[Wo88] A. Waibel and others. Phoneme recognition : Neural networks vs. hidden markov models. In *Proceedings of 1988 ICASSP*, pp. 107–110, 1988.

[YBD+92] Orly Yadid, Eliyaha Brutman, Lior Dvir, Moti Gerner, and Uri Shimony. RAMIT: Neural Network for Recognition of Musical Notes. In *Proceedings of 1992 ICMC*, pp. 128–131, 1992.

[Zel91] D. Zeltzer. *Autonomy, Interaction and Presence*. MIT Media Lab, MA, 1991.

[阿部 91] 阿部, 坂口, 中野. 楽器音の学習認識システム. SICE 学術講演会予稿集, pp. 605–606, 1991.

[永田 91] 永田, 安倍, 城戸. 多数センサによる音源波形の推定. 日本音響学会誌, Vol. 47, No. 4,, 1991.

[園元 83] 園元達, 原田忠則, 松島俊明, 大照完, 橋本周司. 鍵盤楽器演奏のための楽譜の自動認識システム. 昭和 58 年度電子通信学会全国大会, 1983. 1495.

[園元 84a] 園元達, 原田忠則, 松島俊明, 金森克洋, 小沼光良, 大照完, 橋本周司. 高速楽譜認識用マルチプロセッサシステム. 昭和 59 年度電子通信学会総合全国大会, 1984. 1552.

[園元 84b] 園元達, 原田忠則, 松島俊明, 金森克洋, 小沼光良, 上杉明夫, 二村祐地, 橋本周司, 大照完. 高速楽譜認識システム. 第 2 回日本ロボット学会学術講演会, 1984. 3303.

[園元 85] 園元達, 原田忠則, 松島俊明, 金森克洋, 小沼光良, 上杉明夫, 二村祐地, 橋本周司, 大照完. 演奏を目的とした楽譜の自動認識・処理システム. 日本音響学会 音楽音響研究会, Vol. MA84-22,, 1985.

[加藤 86a] 加藤博一, 広野淳之, 井口征士. 規則性を重視した楽譜認識システム. 昭和 61 年度電子通信学会全国大会, 1986. 1603.

[加藤 86b] 加藤博一, 井口征士. 音楽理解システムにおけるピアノ楽譜の自動認識. 情報処理学会第 33 回全国大会, 1986. 3N-9.

[加藤 88] 加藤博一, 井口征士. 小節単位処理に基づいたピアノ楽譜の自動認識. 電子情報通信学会論文誌, Vol. J71-D, No. 5, pp. 894–901, 1988.

[宮尾 89] 宮尾秀俊, 江島俊朗. 楽譜の自動認識. 電子情報通信学会信越地区支部大会, volume 16, 1989.

[宮尾 90a] 宮尾秀俊, 江島俊朗. 印刷楽譜の自動認識. 1990 電子情報通信学会春期全国大会, volume D-470, 1990.

[宮尾 90b] 宮尾秀俊, 江島俊朗, 小谷一孔, 宮原誠. 印刷ピアノ楽譜の自動認識. 信学技報, Vol. PRU90-74,, 1990.

[宮尾 92] 宮尾秀俊, 江島俊朗, 宮原誠, 小谷一孔. 楽譜の規則に基づいた印刷ピアノ楽譜の記号認識. 電子情報通信学会論文誌, Vol. 75-D-II, No. 11, pp. 1848–1855, 1992.

[金森 92] 金森, 平井, 堤, 弓馬, 新美. 即興演奏からのコード情報による曲想の抽出. 音楽情報科学研究会資料, Vol. 37, No. 2, pp. 11–20, 1992.

[原 83] 原, 井口. 複素スペクトルを用いた周波数同定. 計測自動制御学会論文集, Vol. 19, No. 9, pp. 718–723, 1983.

[原田 84] 原田忠則, 園元達, 松島俊明, 金森克洋, 小沼光良, 大照完, 橋本周司. 高速楽譜認識用専用ハードウェア. 昭和 59 年度電子通信学会総合全国大会, 1984. 1553.

[高見 89] 高見, 片寄, 井口. ピアノ演奏における演奏情報の抽出. 電子通信学会誌 D-II, pp. 917–926, 1989.

[寺尾 91] 寺尾晋. 独立信号の波形分離. 大阪大学基礎工学部卒業論文. 大阪大学, 1991.

[秋田 91] 秋田, 増山. ニューラルネットを用いた自動採譜の試み. 音楽情報科学研究会予稿集, pp. 47–52, 1991.

[小沼 85] 小沼光良, 松島俊明, 金森克洋, 橋本周司, 大照完. 楽譜の属性記号の高速自動識別. 昭和 60 年度電子通信学会総合全国大会, 1985. 1616.

[松島 85] 松島俊明, 金森克洋, 大照完. 楽譜の自動認識システム (wabot-2 の視覚系). 日本ロボット学会誌, Vol. 3, No. 4, pp. 354–361, 1985.

[新原 84] 新原, 今井, 井口. 歌唱の自動採譜. 計測自動制御学会論文集, Vol. 20, No. 10, pp. 940–945, 1984.

[青山 82a] 青山宏, 棟上昭男. 印刷楽譜の自動読み取り. 画像電子学会誌, Vol. 11, No. 5, pp. 427–435, 1982.

[青山 82b] 青山宏, 棟上昭男. 印刷楽譜の自動読取り. 信学技報, Vol. PRL82-5,, 1982.

[青山 84a] 青山宏, 遠藤幸一郎, 棟上昭男. 楽譜読取りシステムの高機能化. 情報処理学会第 29 回全国大会, 1984. 3M-10.

[青山 84b] 青山宏, 棟上昭男. 楽譜読取りにおける五線除去の一手法. 情報処理学会第 28 回全国大会, 1984. 1N-6.

[倉富 93] 倉富, 片寄, 井口. 異種楽器の採譜のための特徴量に関する検討. 第 4 6 回情報処理学会全国大会, 1993.

[村尾 92] 村尾忠廣. クロージャーの客観的測定に基づく構造音の抽出について. 音楽情報科学研究会夏のシンポジウム予稿集, pp. 67–72, 1992.

[中村 79] 中村泰明, 進藤正雄, 井口征士. 音楽情報の入力とデータベースの作成. 信学技報, Vol. PRL78-73,, 1979.

[二村 87] 二村祐地, 松島俊明, 大照完, 橋本周司. 楽譜印刷の版下作成を目的とした楽譜自動認識. 電子情報通信学会 創立 70 周年記念総合全国大会, 1987. 1497.

[尾上 79] 尾上守夫, 石塚満, 坪井邦明. 楽譜の自動読取の試み. 昭和 54 年度情報処理学会全国大会, pp. 491–492, 1979.

[富塚 88] 富塚, 白川, 五十嵐. 音楽情報処理システム pshyche と一、二の試み. 人工知能学会全国大会予稿集, pp. 389–392, 1988.

[片寄 88] 片寄, 今井, 井口. 音楽における感性情報抽出の試み. 人工知能学会誌, Vol. 3, No. 6, pp. 748–754, 1988.

[片寄 90a] 片寄, 井口. 知的採譜システム. 人工知能学会誌, Vol. 5, No. 1, pp. 59–66, 1990.

[片寄 90b] 片寄, 福岡, 井口. 音楽解釈システム MIS における演奏ルールの抽出について. 情報処理学会前期全国大会予稿集, 1990.

[片寄 91] 片寄晴弘. 音楽感性情報処理に関する研究. PhD thesis, 大阪大学基礎工学部, 1991.

[片寄 93] 片寄, 村尾. 音楽解釈システム MIS：イデオストラクチャの表現. 情報処理学会全国大会予稿集, 1993.

[柳田 81] 柳田, 角所, 植田, 野村. 一般逆行列を用いたカクテルパーティ効果の知覚的検討. 信学技報, Vol. EA80, No. 69,, 1981.

[和気 89] 和気, 加藤, 才脇, 井口. 演奏者の感情を考慮した協調型演奏システム –jasper–. 音楽情報科学研究会夏のシンポジウム予稿集, pp. 43–46, 1989.

第 4 章

インタラクティブシステム

本章では人間と機械 (計算機) が、対話あるいは協調しながら音楽を作り出して行くインタラクティブシステムについて具体的事例をもとに解説する。音楽そのものを創造する計算機応用におけるこの分野は、伝統的な音楽創造上に新しい可能性を開くものとして、特に欧米では積極的に研究が進められており、現在の計算機音楽の主流を占めていると言ってもよい。以下、新世代楽器、音楽制作ツール、伴奏システムの順に述べて行く。

4.1 新世代楽器 (鈴木)

ここでは、伝統的な楽器に加えて計算機技術を用いることにより人間のジェスチャを音楽そのものに変換しようとする (今までにはなかったという意味での) 新世代楽器について解説する。

4.1.1 RADIO DRUM

RADIO DRUM

Max Mathews らによって開発された打楽器奏法を取り入れた楽器、指揮コントローラである。ここでは [BMS89] をもとに解説する。他のより詳細な文献として 1989 年に国立音大で実演を交えて発表が行われた [Mat89] がある。

概説

RADIO DRUM は電磁波を検出してそこから音楽的情報を取りだすコントローラである。受信アンテナの特別なアレイからなる平面で構成される。送信アンテナは 2 つのドラムスティックの中にある。受信アレイ内には簡単なデータ処理装置があり、ドラムスティックの 3 次元 X、Y、Z 座標を連続的に測れるよう設計されている。ドラムは 2 つのスティックが違う周期で操作されても同時に位置の検出ができる。ドラム面は横 12 インチ、縦 8 インチの大きさを持つ。Z 方向は表面から約 6 インチまで位置の検出ができる。スティックが表面に近い場合は X と Y の精度はその最大値の約 1 パーセントである。ドラムは 2 つのプログラム、すなわち Miller Puckette の Patcher と Max Mathews の指揮者プログラムによって動作する。RADIO DRUM はトリガーをかける装置として、あるいは連続した X、Y、Z の量を用いたコントローラとして使われる。音楽的には打楽器的な即興演奏や伝統的な声楽曲の伴奏に用いられて成果を上げている。

ハードウェア構成

基本構成は受信アンテナを持つ長方形のドラム面と、送信アンテナを内蔵している 2 つのドラムスティックからなっている。(図 4.1参照)

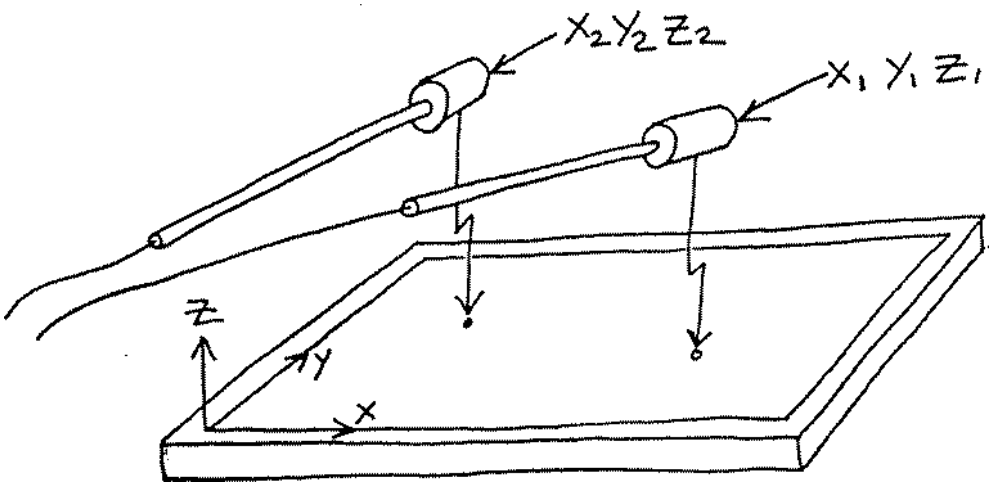


図 4.1: RADIO DRUM 全体図

受信表面では、単純なデータ処理回路でスティックの 3 次元 (X、 Y、 Z) 座標が検出できるよう設計されている。座標読み取り精度は前述の通りだが、Z が増加すると X と Y の精度は減少する。X と Y の測定値は線形性をもっているが Z のそれは非線形性をもっていて、送信機の電磁波の強度とドラム面からの距離によって変わる。ドラム内にはマイクロプロセッサが載っていて信号処理の前段を行い、データをまとめて、RS232C、MIDI、あるいは 2 つのスティックの 3 軸のアナログ値の情報にして外部に送り出す。

使用方法

RADIO DRUM は前述の 2 つのプログラムで動いている。ドラムからは非常に多彩で可能性を持った情報が送られてきて、それを計算機のプログラムによってシンセサイザに対し多様な制御信号を与える事ができる。それらの情報はまだ開拓の余地が残ってはいるが、重要なクラスとして 2 つがある。一つは打撃のトリガーの検出と、もう一つは X、 Y、 Z の連続量の検出である。トリガー検出のアルゴリズムは十分に調べた上で指揮者プログラムによって良好に動作している。ドラム面上に 2 つの目に見えないトリガー面を想定している。表面に近いヒット面とその上にあるリセット面である。(図 4.2参照)

ドラムが初期設定後では、スティックがヒット面に到達するとトリガーが発生する。スティックがリセット面より上に上がるとドラムはリセットされる。ヒット面とリセット面の間では、以前のイベント状態をトリガーするヒステリシスを持っている。打撃の強さを測る良い方法は、スティックがヒット面に近づいたときのスティックの速度を測ることである。この速度はスティックが 1 あるいは 2 ミリ秒に動いた Z 軸の距離を測って求めることができる。

X、 Y、 Z の連続量はシンセサイザの音色のモジュレーション、すなわちスペクトルやエンベロープの制御やビブラート等に使うことができる。そのような音色のモジュレーションはライブ演奏の場合に劇的な効果を上げる。伝統的なコントロール装置であるキーボード、フットペダルやノブ等ではできないシンセサイザの音色の制御に役に立つ。このことは広い意味で従来の音響合成ハードウェアにおけるリアルタイム制御の制限から解放されることを意味している。

RADIO DRUM は即興演奏における打楽器のコントローラとして有効であり、打楽器のための新しい作品のコントローラとして、あるいは伝統的な声楽曲の伴奏や指揮のためのコントローラとして実際に使ってみてその効果が実証されている。

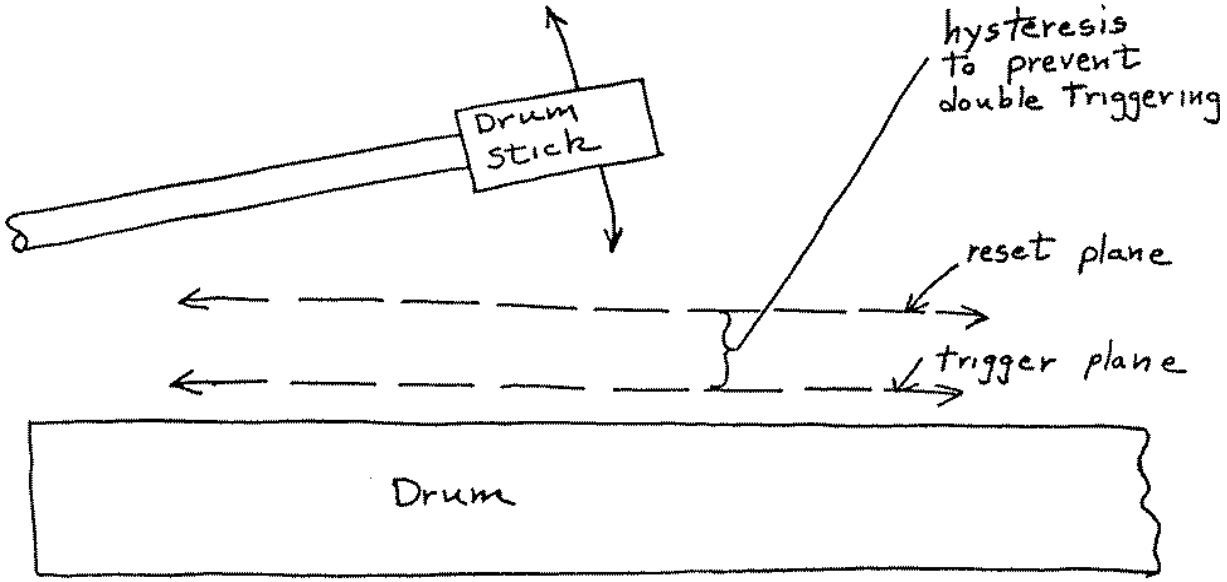


図 4.2: トリガー面

4.1.2 MIDI BATON

概説

MIDI BATON はライブパフォーマンスにおいて指揮者が持つバトンと同等の働きをする MIDI コントローラとして使われる。

以下、[KG89] をもとに解説を行う。

MIDI 制御シンセサイザやペリフェラルは、オペラやミュージックシアターにおけるライブ性とシークェンスパフォーマンスを結び付けたようなリアルタイム電子音響伴奏システムにおいて強力に役に立つ。そのようなシステムはフルオーケストラと比較しても、より広い音色の選択と音程とリズムのコントロールの可能性を、作曲者が書くスコアに与えることができる。しかしながら機械的なクロックに同期したシステムでは融通性に欠けてしまう。そこで CAMTeC(クイーンズ大学) では MIDI BATON を開発した。その応用目的はミュージックシアターで使うことであり、MIDI タイミング情報を電子音響装置に与えて人間のパフォーマーと視覚的なタイミングの協調を行う事にある。バトンタイプのコントローラで指揮者が MIDI 機器に指示を与える。技術的問題も残っており、バトンの物理的な動きから発生するタイミング信号の信頼性をより向上させる必要がある。

システム

MIDI BATON は 2 つの部分から成っている。コントローラ部とそのインターフェース部である。(図 4.3参照)

・バトンコントローラ部

バトン内のメカニカルセンサがある方向に加速度を受けると電気信号を発生する。図中のように真鍮のチューブとその中のスプリングワイヤで吊られた金属の重しから成る。指揮者が振るようにこの筒を加速すると重りが動いてコンタクトに近づき、インターフェース部にトリガーをかける。小さい素早い振りで反応したいが、センシティブ過ぎても困る。はじめは敏感すぎた。

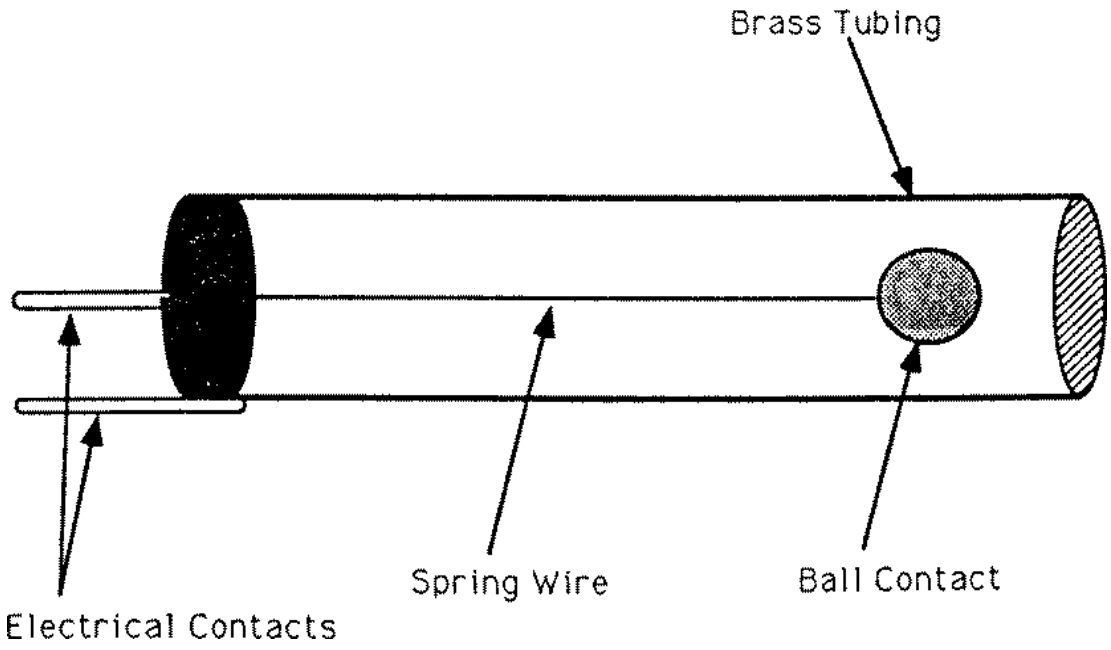


図 4.3: MIDI BATON

それでスプリングの動きを制限するために液体を入れることによって信頼性を上げることができた。

・インターフェース部

信号は約 833Hz(1.2 ミリ秒) でインターフェース部に送られノイズ除去を行い、同期ユニットに送られる。一旦初期化されると一定のパルスを供給する。同期ユニット (シンクロナイザ) は以下の量を調整し決定する。

1. WINDOW : 入力があったビート間の時間量の決定
2. MASK : トリガー間の最小時間のセット
3. DAMPING : テンポ内の自然な変化への反応の度合いのセット

テンポ内の潜在的な変化の度合いを減らすためにシンクロナイザに届く前にノイズを減らす努力をしている。

・シンクロナイザ

MIDI BATON のインターフェース部のパルスは商用のライブシンクロナイザのアナログ入力に送られる。それはバトンの信号を読んでテンポの調整を予想して、それに応じて MIDI シーケェンサを進める。システムの出力は、タイムベースで 24、48、96、384 クロックを含み、ローランドとコルグの DIN シンク、MIDI、メトロノームクロックとスタート / ストップに対応する。それらはまたタイムコマンダのための SMPTE/MIDI タイムコードにも対応可能である。

操作法

この MIDI BATON とタイムコマンダシステムはテンポの加速、減速、すなわちあるビートから 70 パーセント以上ものテンポの突然の変化にも対応する。システムははじめはデフォルト

のテンポでスタートする。フットペダルで指揮者はシークェンサのスタート、ポーズ、再スタートができるが、今はバトンについているマイクロスイッチでできるようにしている。タイムコマンダは再スタートの位置を以下のように選べる。

1. シークェンサのはじめ
2. シークェンサの止まったところ

ライブでは (b) が有効である。

結論

2 つの部分で改良が考えられている。([KW91] 参照)　一つはスタート / ストップの方法であり、一つはバトンの物理的な独立、すなわちケーブルなしのラジオコントロール化である。今回開発された MIDI BATON を用いてシンセサイザを使ったライブ演奏での突然のあるいは柔軟なテンポの変化に対応するシステムを構築することが可能となっている。

4.1.3 VIDEO HARP

VIDEO HARP は複数の指の動作 (ジェスチャ) を検出する光学スキャン楽器である。以下、[RM88] をもとに解説を行う。センサのメカニズム等の詳細は [RM90] を参照のこと。

概説

指先のイメージ画像を検出して、それらの位置と速度を導き出して動作する。得られた情報は MIDI に翻訳されて音響作品におけるシンセサイザ等をドライブさせる。翻訳変換過程はプログラマブルで、楽器としての異なったアーティキュレーション奏法にすることができる。VIDEO HARP はボーイングとドラミングモニタを用いて、ハープあるいは鍵盤楽器のようなジェスチャによって演奏される楽器と言える。

VIDEO HARP は人間のジェスチャを音楽にマッピングする。そういう意味では、前述の Max Mathews の RADIO DRUM と似ていると言える。演奏者から見ると VIDEO HARP は 2 つの概念から成っている。1 つは演奏者が実際にジェスチャを行う物理的な楽器である。もう 1 つはジェスチャを MIDI に変換するジェスチャマッピング装置である。以下これらについて説明していく。

物理的な楽器の形状

VIDEO HARP の外形を図 4.4に示す。

物理的には VIDEO HARP は 1 辺が 30cm 程度の 2 つの等しい三角形の透明なアクリル板からなっている。2 つのアクリル板 (以後は単に板とする) は約 15 度の角度を持って根元でつながっている。ネオン管が板の外側に沿って取りつけられている。その反対側には光を反射する鏡が取りつけてある。板の間の空間には光センサとネオン管に電力を供給する装置がある。

この楽器は立ったり座ったりして演奏することができる。立つ時はひもを首にかけて鏡が腹部に来るように楽器を垂直にする。演奏は、左手は左の板の上、右手は右の板の上で手首を固定し指先を触れるようにして行う。VIDEO HARP はスタンド上に垂直に置くこともできる。さらに面白いことに、この楽器はスタンド上に水平におくこともできる。こうすることにより鍵盤やドラム奏法と同じように演奏することができる。水平に複数の VIDEO HARP を置いて多様な使い方をすることもできる。たとえば 6 台の VIDEO HARP を演奏者の周囲に 6 角形に並べ同時に演奏することも可能である。

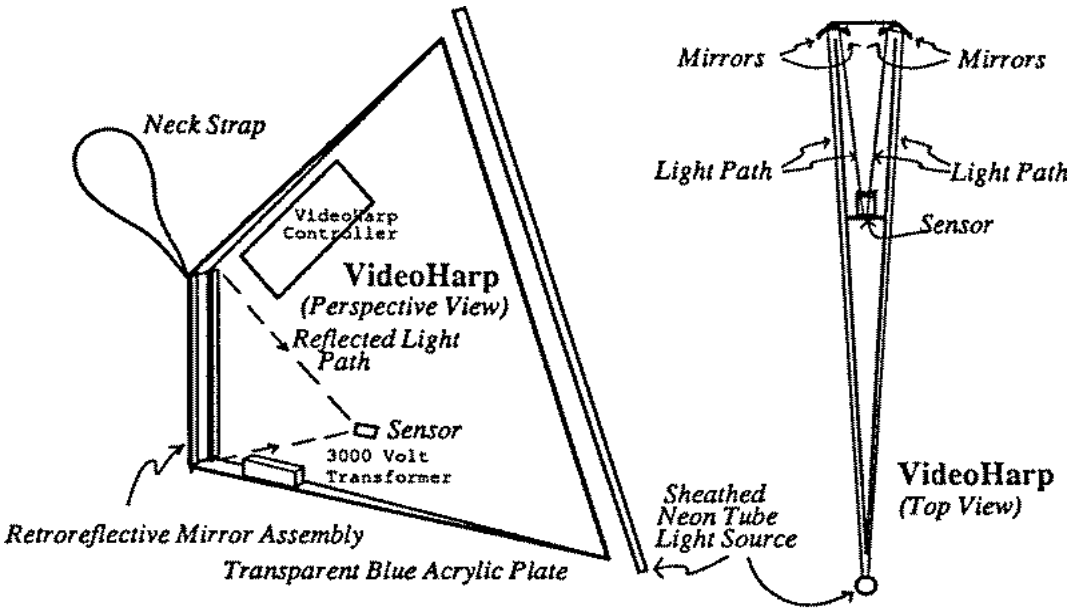


図 4.4: VIDEO HARP

2 つの三角板が分離しているのは以下の理由である。ネオン管は三角形の底辺に沿ってあり、センサは反対側にある。鏡の目的は光のパスを折りたたむことにある。その結果 1 つのセンサで両方の板を通過する光のパスを検出できる。さらにこうすることによってその他の付加的な電子装置も中に置くことができるようになる。

センシング操作の説明をするのに、はじめは 1 つの板の側からはじめる。通常は、センサは何も遮られていない光束をネオン管から見ている。演奏者が自分の指を板の上に置くと指はその光束を部分的に遮り、センサに影のイメージを作る。センサは 1 つの連続的な光束をもはや見てはいなく、指の影によっていくつかの光束のセグメントに分解されている状態を見ている。現在のセンサでは全体の視角 60 度に対して 1/4 度程度の解像度をもっている。ここで表せる角度のことを ray list と呼び、図 4.5にそれを示す。

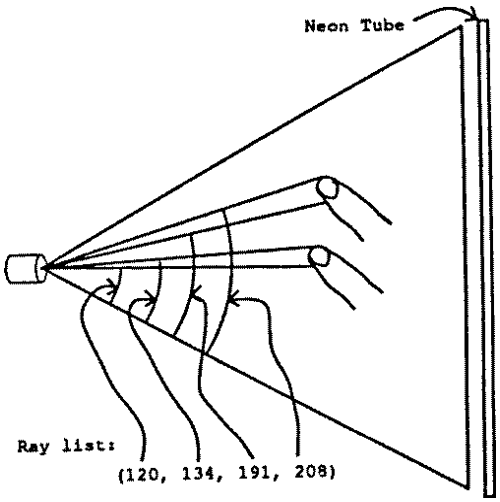


図 4.5: Ray List

指の厚さ (幅) から生じる両端の角度が現れるが、その角度は最も近い 1/4 度になるよう計算され、その大きさによってどのくらい指がセンサに近いか判断している。このようにして影から得られる角度と厚さを測ることにより、指の板上でも位置を求めることが可能である。現在の VIDEO HARP は 2 つの板の上の 1 つの平面をそれぞれモニタしている。ray list は各々の面で 1 秒間に 30 回 (30Hz) でセンスされる。この程度でも十分ではあるが、より速いセンシングができればもっと反応の速い楽器にすることができるであろう。

センサ自身は 1 つの平面以上を検出できる能力を持っている。新しい版のソフトウェアでは各々の板の上の 2 つの平面をセンスできる。内側の面は板から 1.125 インチで、外側は 1.25 インチである。外側から内側の面に指が移動する時間を計算することにより、Z 軸方向のベロシティ (すなわち打撃の強さ) を求めることができる。ただし 30Hz の ray list の検出では十分有効なベロシティを求めることはできない。実際最もゆっくりな指の動きでも 2 つの面を横切る時間は 33 ミリ秒より短い。十分なダイナミックレンジを得るためには少なくとも 1 桁 (3 ミリ秒以内) のスキャンレイトにする必要があると考えられる。また独立した指を見るためには、指のイメージはセンサに対してオーバーラップしてはいけない。オーバーラップの検出は指の本数を数えることによって行われる。

ジェスチャマッピング

ここでは ray list からいかにして MIDI コードにマッピングを行うか解説する。

(a) ハードウェア

センサはコントローラ装置に接続される。その装置は MC68008 プロセッサと 128KB RAM、タイマ、XILINX の LCA を載せた小さな基板からなっていて、三角板の間のセンサの後方に取りつけられている。そのコントローラからはリボンケーブルで、外部の IBM-XT のスロットと接続されている。XT からはローランドの MPU401 を用いて MIDI を出力している。

ジェスチャマッピングはコントローラ部と XT 部で、分かれて行われる。コントローラの仕事はセンサからのデータを読み、ray list に変換してその情報を XT に送ることである。XT はそれよりも高いレベルの解読を行い ray list から MIDI に変換して出力する仕事を行う。

(b) オブジェクト

ジェスチャマッピングプロセスの最小のステップは、ray list をオブジェクトリストに変換することである。オブジェクトという用語は、ここではセンサから見える 1 本の指を記述する属性値の集合と定義される。

1 つのオブジェクトは、タプル (s,θ,t,time,z,uid) で表現される。

- s はオブジェクトが表される VIDEO HARP の左右どちらからのサイドを表す。
- θ はセンサと板の底で作るオブジェクトの中心角度である。その値は 0(底) から 255(頂上) である。従って各々の値が 1/4 度に相当する。
- t はオブジェクトの厚さに相当して、同じユニットの θ と同一になる。t の範囲は 1 から最も厚い 255 まで存在しうる。
- time は内側の面を最初にオブジェクトが横切った時間である。
- z はオブジェクトの方向を示す一連の情報であり、以下がある。

  In: オブジェクトの出現
  Out: オブジェクトの消滅
  Split: 2 本の指で交互にタッチされた状態

Merged: 2本以上の指が重なった状態
Existing: オブジェクトが視界内に存在し、θ と t の値が最後のオブジェクトストから変更されている状態

- uid はユニークなオブジェクトの識別子であり、オブジェクトが視界内にある間にオブジェクトを識別するために使われる。

1つの板に対して1つの面に対応する2つの ray list をオブジェクトリストに変換するアルゴリズムを述べる。2つの板は s が異なるということだけで分離して考えることができる。各々のサイドで新しい ray list と以前の ray list を求めるアルゴリズムが考えられる。新しい ray list がセンサから入力される前に、以前のオブジェクトリストは新しいオブジェクトリストは何であるかを予測するのに使われる。各々のオブジェクトでは、現在の位置と厚さはオブジェクトの新しい位置と厚さを予測するのに使われる。新しい ray list はその後入力され、各々のイメージに対して θ と t が与えられた部分的なオブジェクトリストになる。その後予測されたものと新しいオブジェクトが比較される。各々の予測オブジェクトでは、その θ を中心に持ったあるウィンドウ(現在では予測された t の3倍)があり、このウィンドウに落ちる新しいリストからのオブジェクトは同じ指を示すと見なされる。

このようにマッチングを与えて新しいオブジェクトが計算される。予測されたオブジェクトにマッチングしない新しい ray list からのオブジェクトは In として定義される。もし新しい ray list からの複数のオブジェクトが予測されたオブジェクトの中の1つのオブジェクトにマッチしたら、新しいオブジェクトは全て split される。同様に予測されたリスト中の1つ以上のオブジェクトにマッチした新しい ray list からのオブジェクトは Merge される。1つの予測オブジェクトに排他的にマッチした新しいオブジェクトは Existing である。唯一のはっきり決まらないケースは、2つ以上の新しいオブジェクトが2つ以上の予測オブジェクトにマッチしたときである。これを解決する方法として、あいまいさがそれ以上存在しなくなるまで現在の新しいオブジェクトと予測オブジェクト間の最も大きな距離でマッチングを同期的に消去する方法を取っている。

(c) 領域、変数と MIDI イベント

一旦オブジェクトリストが計算されると次のステップは各々のオブジェクトを領域に割りつけることである。直感的には VIDEO HARP の板の領域をあるオブジェクトから MIDI データに変換することであろう。技術的には、ここでの領域は s の選択(左右)と θ と t の範囲内である。VIDEO HARP が演奏される場合、たくさんのアクティブ領域が存在する。オブジェクトは出現しては動き回り、消滅して、それらは MIDI イベントとして出力される。通常ユーザは同時に演奏されるたくさんのオーバーラップしていない領域をセットアップして、それらを VIDEO HARP のプリセットとしてまとめてグループ化できるであろう。演奏中にユーザは簡単に VIDEO HARP のプリセットをスイッチで切り替えることにより、楽器の性格を容易に変えることができる。

割りつけられた領域は MIDI に対して特別なマッピングを施される。そのようにするために各々の領域では多くの変数が計算される。2種類の変数がある。モノフォニックとポリフォニックである。ある領域内では各々のモノフォニック変数の4つのインスタンスのみが存在する。その領域内で起こる各々のオブジェクトに対する各々のポリフォニックな1つのインスタンスが存在する。どちらの場合でも変数のセットはプログラム可能である。ユーザは発生を望む変数や、その変数でいかにして MIDI イベントにトリガーをかけるか等を指定できる。領域の型は C で記述され、1つの型の記述に約 100 から 200 行を要している。

応用

センサ入力を MIDI イベントにマッピングする方法は、多種のジェスチャを多種の MIDI コードに変換するために汎用性を持って記述される。しかし、この汎用性のために記述が複雑になっ

てしまう。エンドユーザにとっては、この複雑さは好ましいことではない。そこでユーザが領域を種々のプリセットができるよう考えた。

以下に最初の版で試みた領域のプログラミングの例を示す。ここでは直感的に理解できる例を取りあげたが、実験的に違った試みをすることが可能である。

(a) キーボード領域

ここでは鍵盤を弾くテクニックで VIDEO HARP を演奏できるよう設計されている。領域内の演奏者の指である音の発生を引き起こす。この領域では、θ は MIDI ピッチに、初期の t は MIDI ベロシティ、その後の t は MIDI アフタータッチにマップされる。さらに uid は MIDI チャネルにマッピングしている。ある指の基準ごとに MIDI ピッチベンドを送ることも可能である。キーボード領域ではいくつかのモノフォニック変数が有効であるが、主としてポリフォニックとして使われる。たとえば指の厚さを MIDI モジュレーションホイールや MIDI チャネルプレッシャーコードにマップできる。

(b) ボーイング領域

ボーイング領域のアイデアは擦弦楽器を弾くシミュレーションから来ている。弓奏を行う手のみがシミュレートされる。他の領域で (あるいは他の MIDI キーボードのようなコントローラで) ピッチの発生を行う。弓の速度と弓のブリッジの近さは、それぞれ時間に関する θ と明らかな指の厚さ t でモデル化される。他の指の属性で付加的なパラメータを使うことができる。この領域は全てモノフォニックである。θ の変化率は MIDI ブレスコントローラやフットコントローラのような MIDI ボリューム制御コードにマップされる。同様に指の厚さ t は他の MIDI コントロールコードにマップされる。もしも 2 本の指が領域内にあるときは、2 本の指の距離で MIDI ピッチホイールやモジュレーションホイールを制御できる。

(c) 指揮領域

ここはボーイング領域と似たような演奏をするが θ の変化量で MIDI クロックをコントロールしようとするアイデアである。これによりシークェンスのテンポが制御できる。ボーイング領域と同じように、他の属性は他の MIDI コードを発生させる。特別には、他の指でシークェンスの開始や種々の MIDI チャネルの相対ボリュームの制御を行う。この方法で、演奏者はリアルタイムで MIDI シークェンスを指揮者のように指示できる。

(d) コントロール領域

この領域では VIDEO HARP をユーザは各々の θ のサブレンジに対して任意の MIDI コードを送ることができる。一般的には、ここでは MIDI プログラムチェンジコードを送る。

問題点と将来の計画

現在のセンサの解像度は 1/4 度である。通常はそれで十分であるが、さらに精度を上げると指の厚さより正確に測れることになる。同様に 30Hz のスキャン周波数はベロシティ検出には十分であるとは言えない。より速いセンサとスマートな検出アルゴリズムの改良が必要である。

将来的には VIDEO HARP を演奏する際にホストワークステーションをなくしたいと考えている。これは MIDI 発生器を楽器の中に組み込むことを意味する。現在の MC68008 をより強力なプロセッサに変え、オブジェクトリストとイベント発生を楽器内で完結させたいと考えている。

現在の版の VIDEO HARP ソフトウェアでは、計算をしていない付加的属性オブジェクトがある。例として解放時のベロシティー (すなわちオブジェクトが視野から離れるときの速度) とオブジェクトの角度 (すなわち外側と内側の面内にあるオブジェクトの θ 値の相違によって計算される板に対する指の角度) がある。将来はこういったパラメータも入れたいと考えている。

最後に VIDEO HARP の操作の中に、より複雑なジェスチャ認識を結びつけたいと考えている。現在ではオブジェクトの動きはかなり直接イベントにマップされている。完全なジェスチャをよりシンプルな動作にマッピングすることは興味深いことである。例として、指を交互に素早くかき鳴らす動作で、アッチェランドやクレッシェンドを指示するようなことである。

結論

VIDEO HARP は光学センスを行い、指の動きを MIDI コードにマッピングする楽器である。マッピングはプログラム可能なので、演奏技術の潜在的な多様さは伝統的な楽器にあるもの以上にたくさんあるといえる。ある音楽を演奏するために作られて来た楽器の場合には、一般的に表現はある可能性の範囲内に限られるが、VIDEO HARP は楽器自体を望みの音楽表現にかなうように仕立て直すことができるのである。VIDEO HARP が作曲家と演奏家の両者に新しい音楽表現の大いなる道を開くことを望んでいる。

4.1.4 HYPERINSTRUMENT

MIT メディアラボの Tod Machover らによって開発が行われている新世代楽器である。以下、[MC89] と [CGN91] をもとに解説を行う。

概説

[MC89] Hyperinstrument(ハイパー・インストゥラメント; 以後 HI と表記) が目指していることは楽器の概念を拡大することである。プロの音楽家の習熟度合いに従って楽器の能力も上がっていくことである。加えてマシンと人間の間の協調関係の適度なバランスを考察して、一人のプロの音楽家でコントロール可能な限界をつくることでもある。

HI の研究の目的は、音楽的な知性を持ったマシンと人間がインタラクティブな演奏ができることである。楽器の技術と知識ベースとインテリジェントな音楽構造を作ることで音楽表現の再構築ができる。現在の HI は、2 つのコントローラ (Yamaha KX88 MIDI キーボードとシリコンマレットやオクトパッドのようなパーカッションコントローラ) の並列入力を行っている。MIDI は 2 つの楽器からモニタされて、8MB メモリを持つ Mac II に送られる。Mac 内では開発されたリアルタイム MIDI/LISP 環境 (Hyperlisp で記述) が動いている。一旦データが Hyperlisp に受け取られると、それは分析されて音楽的なイベント情報として MIDI にコード化され、多様なディジタルシンセサイザやサンプラに送られる。

ソフトウェア

Hyperlisp は Mac の Allegro Common Lisp 中で動くオブジェクト指向プログラムである。Hyperlisp は HI プログラマに多くのサービスを提供する。すなわち 1/100 秒分解能のスケジューリング、バックグランドタスク、全 2 重の 2 つの Mac シリアルポートの制御、MIDI in/out の Lisp サポートである。Hyperlisp はタイムタグ順序化をなくした特別なスケジューリングアルゴリズムを使っている。Hyperlisp はまた MIDI ドライバのスケジューリングができ、リアルタイムで計算後 MIDI 出力を行ったり、ディレイをかけることができる。

HI プログラミング環境は多くのソフトウェアモジュールから成っており、それらを違った組み合わせでリンクすることにより一つの楽器を形成する。たくさんのプログラマは異なった HI

オブジェクトを同時に開発できて(名前の管理、冗長さのチェック、相互関係の衝突チェックを気にせずに)、同じマシン上で一つ以上のHIを独立に動かすことも可能である。

もっとも役に立つモジュールは、再生/記録のオブジェクトと予めシークェンスされたデータのリアルタイム操作オブジェクトである。これはスコアオブジェクトと呼ばれ、これらはシークェンスMIDIデータの集合と全ての手続きと操作(それらはシークェンスプレイ、テンポ変化、ピッチとアクセントの修正を行う)と考えられる。

グリッドオブジェクトはスコアオブジェクトのように振る舞うが、それに加えて他のスコアとの同期機能を持っており、その拍を用いて演奏できる。グリッドの使い方として、一つあるいはそれ以上のスコア(実演奏によってトリガーされた)を共通のリファレンスに同期できる。他の使い方は、ライブ量子化で次の拍までイベントを遅らせる。グリッドは最後の拍が起こったときと次の拍が起こるだろうときに計算を行わせることができる。スコアがグリッドに同期したとき、グリッドはスコアを加速したり減速してスコアの拍はグリッドの最も近い拍に整列させる。その整列のための時間は調整できて、同期化は望むようにスムースに行える。

メタスコアクラスの操作では、7つ以上のスコア入力を行い、他のスコアを出力として生成する。例としてスコアスプライシング操作は二つのスコアを取ってきて、1番目に2番目を付加したような3番目のスコアを生成する。実際、3番目は単に1番目、2番目と演奏され、新しいデータは生成されない。それでも新しいスコアは有効なスコアであり、それ自身スプライシング操作や他のメタスコア操作の入力として使用できる。この方法は複雑だが、リアルタイム変換が単純なものの組み合わせから生成される。

最近の研究

重要なHIの改良は、Tod Machoverのオペラ「ヴァリス」(1988)で行われた。これらの開発は、打楽器と鍵盤楽器の演奏技術に関して行われ、シリコンマレットコントローラやYamaha KX88、Kurzweil MIDIボードが使用された。以下に、これらに関する重要な点を述べる。

(a) リズムの拡大

種々のリアルタイムマシンコントロールを通して演奏者のリズム技巧と正確さの技術を高めるコンセプトを探究した。最も成功した技術は前述のグリッドオブジェクトで、バックグランドで聞き取れない位の非常に速いリズムの流れを作り出した。グリッドは同時に二つの目的で使われた。一つはキーボード演奏者による自動化されたイベントトリガーの正確なスタートポイントの決定で、二つ目は音程のないドラムのような打楽器で演奏された全ての音が調整可能なグリッドポイントだけで鳴ることである。打楽器奏者はそれゆえアーティキュレーションや音色選択に集中できて、なおかつ他の楽器に完全に同期した正確で驚異的なリズム配列を作り出すことが可能である。

(b) タイムワープ

他のリズムの複雑さの拡大技術は、同様の量子化グリッドを使ってテストされた。グリッドの拍に従わせるというよりは、複雑なリズムパターンが予め作曲され計算機に貯えられた。音程も持っている特別なリズムパターンが鍵盤の上の特別な音程によってトリガーされた。これらのパターンは最も近いグリッドポイントに同期化された。一旦トリガーされるとパターンは一時非同期に、あるいは加速して“ワープ”され、鍵盤のポリフォニックアフタープレッシャーによって再同期化することができる。

(c) シークェンスデータによる演奏

演奏時のジェスチャを用いて、予め貯えられた音楽的な題材をより洗練させる多くの試みがなされた。最も苦心したシステムはキーボードベースのものである。左手の鍵盤のある音が音楽シークェンスをトリガーした。すなわちトリガー音のベロシティーがシークェンスの音量を決め、アフタープレッシャーが 10 個の異なった音色の一つを選んだ。右手はシークェンスのアーティキュレーションに使われた。白鍵のベロシティー値は次のシークェンスのベロシティを決める。従って微妙な音量の調整が可能である。さらに黒鍵では特殊な音色で次のシークェンスを保持することができる。

(d) アルペジオの高速化

ヴァリスの自動アルペジオシステムでは二つの改良点が加えられた。一つは知的サーチアルゴリズムである。これは鍵盤の演奏法によって適当なパターンを選ぶものである。二つ目は Kurzweil MIDI ボードのポリフォニックアフタープレッシャー機能によって開発されたもので、各々のアルペジオノートを独立な音色のアーティキュレーションにすることができる。このポリフォニックアフタープレッシャーの機能はシステムにとって重要な役を果たした。なぜなら多様な音楽表現、特に連続的な音色の変化のような表現を容易なコントロールでマッピングできたからである。

(e) トレモロによる音色形成

HI では打楽器の技術で、音色のような連続的なパラメータを使っている。ほとんどの演奏者がマスタしている音量とスピードの 2 つをはっきりと分離できる 2 つのマレットのトレモロ奏法を選んだ。トレモロのスピードは以下のように音色を決定する。

10 種の音色が使えて、最も遅いトレモロスピードが 1 つの音色を選ぶ。除々にスピードが増すと次の音色が付加されていく。スピードと音色の変化のマッピングをどのように行えばよいか、音色のオーバーラップはどのくらいがよいかの実験がなされた。HI は分離した音楽データを連続的な音楽的変化にする有効なモデルを提示している。

(f) ダブル楽器

ダブル楽器は、分離した物理的コントローラ上での 2 人の演奏からの音楽的ジェスチャを組み合わせることにより一つの演奏を作り出す楽器のことである。この楽器によって従来のようなアコースティックな組み合わせに代わって、新しい種類のアンサンブルが可能になり、演奏者は、お互いのパフォーマンスのジェスチャを音楽的に結び付けることができる。様々な音楽家が音楽のある状態に影響を与えられるよう HI が設計された。しかし両者が全体の音楽的な結果を作るためのアンサンブル内で演奏することは必要である。一つのバリエーションでは、打楽器奏者のあるノートが現在キーボードで弾いている音の音程に再マッピングされるというようなことである。打楽器上の各々のオクターブは同じ音だが違う音色に鳴る。加えてキーボード演奏者はポリフォニックアフタープレッシャーを使ってマッピングの重みを変化させ、ある音が他の音よりひんぱんに現れるようにできる。他の応用では打楽器奏者はあるオクターブの 7 つの白鍵だけを演奏すれはよい。これらのキーはキーボード奏者が多様なハーモニックスの音程の集合を作り出す 7 つの周波数レンジを示している。キーボードのアフタータッチは周波数レンジやより多くの音色を加えるために使われる。音色は打楽器奏者の打撃の強さによってコントロールされる。ダブル楽器の奏者は各々の音楽的ジェスチャを伝統楽器のような音になるよう音楽的ジェスチャを関係付ける必要があるだけでなく、他の奏者のジェスチャや音にも関係があることを忘れてはならない。このダブル楽器は複数の奏者が音楽的にリンクしている新しい種類の音楽的アンサンブルを作り出す最初のステップと位置付けしている。

音楽作品の例

最近の HI を用いた音楽作品の例として、Tod Machover の「Towards the Center」(1989/1990) がある。この作品は 6 種類の楽器 (フルート、クラリネット、ヴァイオリン、チェロ、キーボード、打楽器) と指揮者からなる。キーボードと打楽器が、この作品のために特にデザインされた HI で演奏される。全ての複雑な電子音楽は、予めレコーディングされずに HI 上でライブ演奏される。4 つのアコースティック楽器は、HI 経由で制御される複数台のエフェクタ Yamaha SPX-90 によって音色を変えられた後 PA を通して聞かれることになる。

弦楽器を用いた HI

[CGN91] MIDI 楽器ではない “連続音” の楽器である弦楽器、ここではチェロを用いた報告である。試作は MacIIfx をベースに開発された。Mac のペリフェラルとして Digidesign Audiomedia DSP カード、EXOS ハンドセンサー、General Instruments MacADIOS データ収集カード、シリアルインターフェース (MIDI)、特別仕様のハードウェアから成る。プログラミング環境は、Mac Common Lisp オブジェクトシステムである。それらはソートされないスケジューラ、MIDI 管理支援、CLOS オブジェクト (音楽プロセスの外部インターフェース制御、分析、グラフィックス、リアルタイム制御) を含んでいる。

チェロは多くの連続した自由度を持った成熟して反応が良いユーザインターフェースを持っており、電子楽器のリアルタイム制御を探求するには興味深い楽器と言える。初めに奏者の制御入力の役割としての観点から、弦の物理をレビューした。そしてチェロの弓の位置と圧力情報から得られるジェスチャ情報を抽出した。こららのパラメータを用いて種々のテストをして、センサの精度や信頼性のトレードオフの考察を行った。

プロのチェロ奏法からの音楽的ジェスチャの解析は、ピッチトラッキング技術とジェスチャトラッキング技術を用いて行われた。1990 年から 1991 年にかけて、チェロ奏者のヨー・ヨー・マが MIT メディアラボの HI 研究に協力した。目的の一つは、マをモニタしてより洗練されたジェスチャのセンシングを行うことであった。マと他の 2 人のチェロ奏者によって演奏されている間、データは HI ジェスチャセンサ (親指と人差し指の圧力データとボーイングを行う手首の角度の情報の記録) による情報が集められ、ディジタルレコーディングされたオーディオデータに同期して記録された。データは違った実験状況下で、以下の演奏パラメータを使って解析された。すなわち、ルバート、レガート、ビブラート、チューニング、タイミング逸脱とボーイングである。音響とジェスチャのデータの解析結果としては、演奏する音楽的意図の認識のマッピングはある演奏パラメータの規則支配によって特徴づけられ、これらの規則は奏者と楽器のクラスとして一般化され得るということであった。

4.1.5 まとめ

人間のジェスチャを楽器の演奏動作として捕らえるインタラクティブシステムを、新世代楽器としての具体例を論文をもとに示してきた。ここでの特徴的なことは、それらの演奏には高度なテクニックが要求されるということである。すなわち、ボタン一つで複雑な音楽を (勝手に) 計算機にやらせるのではなく、あくまで人間が高度に表現したい音楽的思考を忠実に反映させる努力がなされているということである。そのために人間は、これらの「楽器」を使いこなすためにシステムの熟知と高度な訓練が要求される。そういった意味では、これらの新世代楽器は、明らかにプロの演奏家を指向していると言えるし、いわゆる伝統楽器の延長線上に位置しようと意図されていると言えよう。言葉を変えると、今までの伝統楽器で表現できる範囲を超えようとする努力とも言える。一般の伝統楽器では、物理的制限から限られていた音楽表現をより広げようとする、いわば楽器の概念の拡大化を目指しているということであろう。

以上のことが可能になってきたのは計算機科学の進歩であることは言うまでもなく、そうした技術を背景に、今まで考えられてきたいわゆる「楽器」の範疇がさらに広がろうとしている。楽器というと普通はピアノやギターのように、演奏部と発音部が一体の「物」として明確に目に見え定義できるものを思い浮かべる。ところが計算機と電気的な音響装置を使うとそれらが、必ずしも一体化する必要はなくシステムとして「楽器」が定義されるということが起こってくる。ここで取りあげた「楽器」は、まさにそうしたシステム化された形で、ジェスチャ入力部、演算処理部、音源部といった全体ではじめて一つの楽器と呼べるようなものである。

最後にもう一つ私見を言わせてもらえれば、音楽を作りだす道具としての「楽器」というのは、やはりそれにふさわしい姿・形をしていて欲しいと思う。電子的手段と計算機を使えば、形は自由に設計できるであろう。しかし、長い年月をかけて作られてきた楽器というものは、音を出すメカニズムの合理性と同じくらい美しい形をしていると思う。

長い時間をかけて培われてきた楽器に、計算機技術を応用することによって新世代楽器が加わることにより、音楽の世界はより豊かになるであろう。

4.2 音楽制作ツール

前節であげたリアルタイム演奏を目的としたいわゆる「楽器」としてのインタラクティブシステムの他に、音楽制作をやはり人間と計算機の対話で協調して行うシステムが研究開発されている。この節では、グラフィック操作で音楽を作っていく UPIC システム、オブジェクト指向でユーザが自由に MIDI パッチを作っていける MAX、IRCAM で開発された NeXT ベースの音楽ワークステーションの ISPW について順に解説していく。

4.2.1 UPIC (鈴木)

概説

UPIC(Unite Polyagogique Informatique du CEMAMu) は、ユーザによって音響や音楽的素材を直接に扱えるよう設計された計算機ベースの音楽制作システムである。ユーザがグラフィックタブレットに、直接必要なパラメータや音の情報を書き込み、それらがすぐにホスト計算機によって処理されて音に変換される。以下、初めに [Loh86] をもとに歴史と機能の説明を行い、続いて [MRS90] と [MRS91] をもとに新 UPIC システムについて簡単に触れる。

UPIC の歴史

1954 年に Iannis Xenakis が発表したオーケストラ作品「Metastasis」において、図形楽譜から音楽への翻訳が試みられた。Xenakis はこれをさらに発展させて、伝統的な記譜法の次を行くものとして、音楽的思考をより汎用性があり誰にでも解読できる方式を確立しようと考えた。Xenakis は 1966 年に EMAMu を設立し、それが現在広く知られている CEMAMu(Centre d'Etudes de Mathematique et Automatique Musicales) になる。パリに本拠を置く CEMAMu は人間と科学の研究を行う非営利の団体である。Xenakis の目指した CEMAMu の目的は次の 5 点である。

1. 音楽作品と映像作品の領域で科学的アプローチ (数学、音響学、計算機科学等) によって創造を行う研究
2. 上記の研究にあたり、計算機システムや最適化プログラムなどのツールや方法論を確立すること。

3. 上記の目的のために開発されたUPICシステムを用いて作られた研究結果としての音楽作品。作曲家はUPICを用いて触発され、新しい作品を作ることにより現代音楽の発展に寄与する。
4. UPICを広く普及させ、子供たちのような若い世代に新しい創造を行わせること。
5. UPICの誰にでも使える直感的使用法によって、より創造的な活動が行えることを広く知らせて、多くの人に考え方を変えてもらうこと。

1970年代が第1世代UPICで、ミキシング機能を除いて現在と同等な基本的な音を作り出す特徴を持っていた。第2世代はミキシングが可能になり、第3世代では市販的な標準部品を用いて作られ、リアルタイムシンセシスが可能になった。さらに現在ではIBM PC-AT386マシンを使い、よりコンパクト化されている。[MRS90]、[MRS91]

UPICの構成

システム構成を図4.6に示す。

UPICはインタラクティブに図形操作が行えるシステムである。2次元のグラフィックタブレットと電磁ペンが入力装置である。そのグラフィックデジタイザがこのシステムの心臓部で、高さが60cmで幅が75cmである。入力精度は0.25mmである。このデジタイザの右側にはコマンド入力エリアとメモリアクセスのためのいくつかのエリアがあり、タッチペンで操作できる。描いた内容は同時に2つのCRT、すなわち一つはグラフィックディスプレイでもう一つは文字表示ディスプレイに表示される。描かれた図形情報は16ビット計算機システム(現在は32ビット)に入力され処理される。システムには他にDA変換器とAD変換器が付加される。標準サンプリングレイトは50kHzで、AD変換器を介してベートーヴェンの交響曲をもサンプリングできる高機能なシステムを持っている。計算機を動かすソフトウェアは多様なレベルではあるが、総数500本以上になっている。このような複雑なシステム構成をしているが、エンドユーザが使用するには、内部の複雑な構成を知ることなく使えることがUPICの大きな特徴になっている。

ユーザから見たUPICの機能

UPICは容易に扱えるシステムで、2、3時間で基本的な機能はマスターできる。メインメニューは機能ごとに分かれていて、各々の機能はより深いサブコマンドへと分岐していく。全体として自己充足な信号処理システムと見ることもできる。UPIC内の計算機プログラムで音が作られ、計算機内にはプリセット音はない。フーリエ級数シンセシスやスペクトル解析といった機能が用意されている。作曲レベルとして以下の2つについて解説する。

(a) マイクロコンポジションレベル

スクラッチから音楽作品を構成していくために、一般的にはマイクロコンポジションレベルから始める。使いたいと望む音の構成要素を定義する。3種の基本的な音色の構成要素を使ってデザインすることができる。すなわち(1)波形、(2)ダイナミックエンベロープ、(3)1と2から成るmusical waveformと呼ばれる要素である。

ボード上のOKスペースに触れると、グラフィックタブレット上に波形がトレースできる。トレース結果は同時にCRT上で見ることができる。時間軸は計算機によって自動的に接続される。結果を聞くにはピッチを決められたタブレット領域から入力することによって対応する音が発音される。ダイナミックエンベロープは個々の音ごとに与えらるのはもちろんであるが、マイクロコンポジションレベルにおける音楽全体にもかけることができる。musical waveform機能は基本的には編集操作であり、どのコンポジションレベルでも応用できる。グラフィックタブレッ

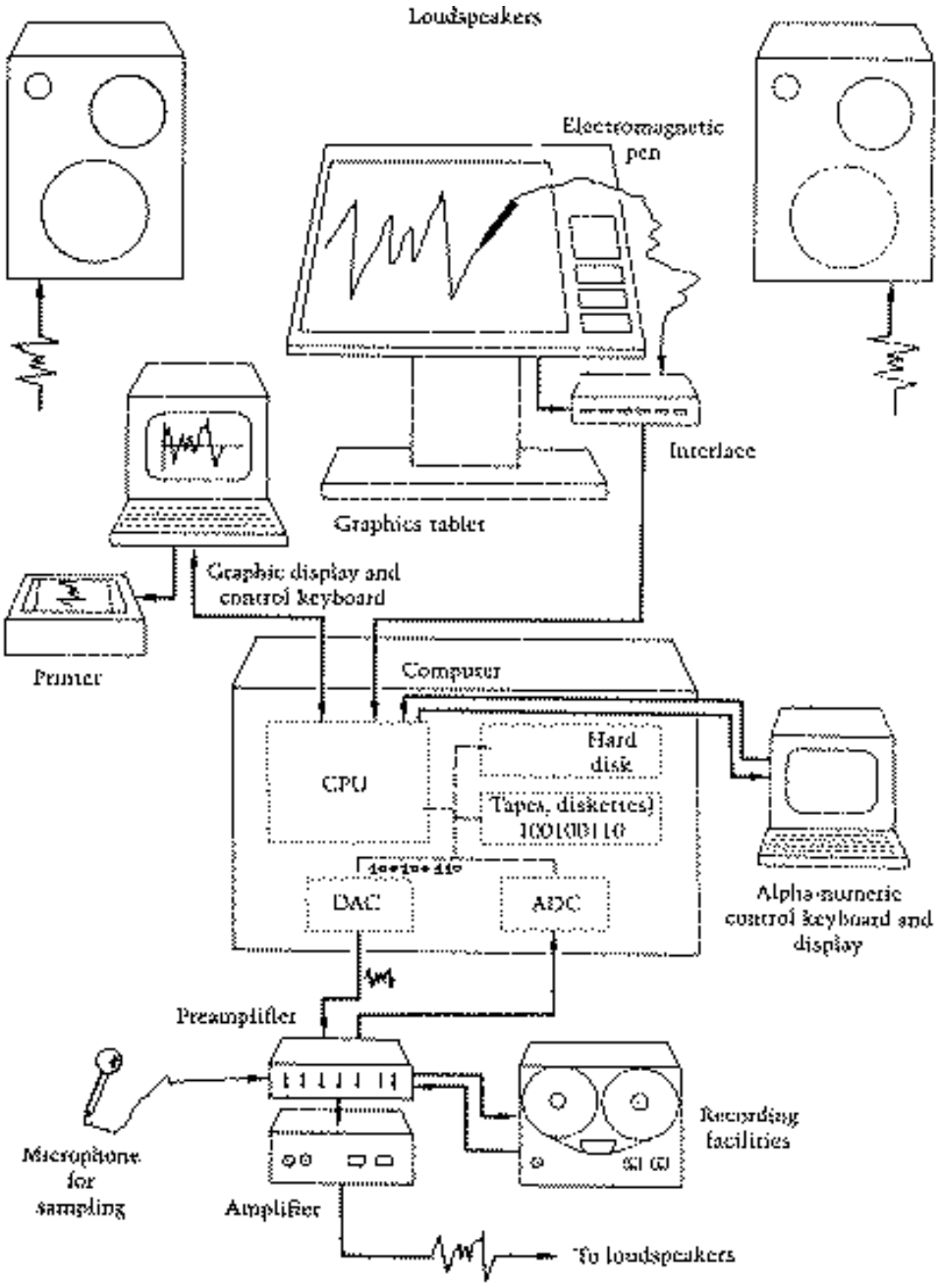


図 4.6: UPIC 構成図

トに触れて CRT 上の画像のあるセグメントを選んで、挿入、削除等の編集が行える。またセグメントを選んでズームアップができる。必要なら個々のサンプルレベルまで可能である。不要なグリッチ等のノイズ除去も行え、以前に作ってある素材やサンプルされた素材を埋め込むことができる。これらの編集機能はサンプリングマシンとして動かした場合に非常に役に立つ。musical waveform 機能によって前もって貯えられている波形やエンベロープなどの素材を自由に結びつけることができる。

(b) マクロコンポジションレベル

このレベルでは、music page 機能が曲作りをする際に必要なパラメータデザインのために呼び出される。システム内の音楽のページは、あたかも目の前に紙面に書かれた音楽的思考が広

がっているように見える。Xenakis は伝統的な記譜法を用いるようには設計せず、全て図形処理で行えるように考えた。従ってグリッサンドのような情報も視覚的に表すことができる。またこのレベルは全オーディオ可聴領域に対応しており、ユーザは伝統的な記譜法の制約から解放されている。時間軸に対しても、ページは 0.2 ミリ秒から 30 分程度 (ディスクの容量による) の持続時間をプログラムできる。自分のアイデアをボード上にマーキングしながら描いて行ける。タブレット上にペンで直線や曲線や点を描きながら作曲を行っていける。Xenakis はそれらをソニックラインと呼んでいる。このようにページ上に音楽を描き終えると、それをオーケストレーションする。これは前もってデザインしておいた音色素材やエンベロープ素材を、各々の直線や点や図形オブジェクトに割りつけて行く作業である。

あるページは初期状態では時間のパラメータがない。ページが完成するとページの持続時間を与える。UPIC はそれを計算して、その結果は music wave と呼ばれる。普通の紙面の楽譜のように、1 枚のページが 1 つのページに対応し、さらに次のページが連続していく。

ミキシング機能がこのシステムの要になるポイントである。この機能はあらゆる可能なページのオーバーラップやレイヤ化をミリ秒精度の時間軸上で行える。さらにミキシング機能は波形操作レベルのような他のどのレベルでも行える。作曲の道具の中で、この機能は計算機の特質を最も生かした機能であり、実際作曲のどのレベルでも非常に多用されている。多くのサウンドモジュレーションもこの機能を通して行われる。すなわちマイクロコンポジションレベルではよりライブな音を作るために、ディレイやルームリフレクション等の効果音が作られる。

UPIC における考察

UPIC 研究の主目的の一つは、音をアニメーション化して作りだす方法を探求することにある。UPIC は人間と機械が対話をしながら音楽を創造していく一つの解答を与えている。Xenakis の UPIC のアイデアは、彼の一般的なレベルでの現代音楽の創造のためのユニークなアプローチの一つであったが、UPIC が潜在的に持っているアイデアは人間の持っている直感的な音楽的思考を直接反映させることができるといったより汎用性にあると言える。

新 UPIC システム

リアルタイムシステム版の UPIC が 1988 年ケルンの ICMC で発表され、IBM PC-AT386 ベースの新 UPIC システムが 1990 年グラスゴーの ICMC で発表された。[MRS90] さらにこの新 UPIC システムのデモンストレーションが 1991 年モントリオールの ICMC で行われている。[MRS91]

以下は新 UPIC システムの概略である。

新 UPIC システムは、グラフィックスコアエディタとボイスエディタとパフォーマンスシステムを組み合わせたインタラクティブな音楽制作システムであることは旧 UPIC システムと変わりないが、コマンド入力にはマウス入力が取り入れられている。ページの持続時間は 6 ミリ秒から最大 1 時間まで拡張されている。演奏中に音楽家はあるページから別のページへの切り替えが容易にでき、マウスを動かすことにより演奏位置を簡単に制御できる。ライブ演奏の結果はシークェンスと呼ばれ、編集可能なオブジェクトとしてレコーディングされる。同時に 4 ページがウィンドウとして画面上に開かれる。ページ編集コマンドとしては、カット、コピー、ペーストがあり、インタラクティブに編集が可能である。演奏システムとしてはページごとに 16 チャネルのオーディオ出力が定義できる。このページごとにダイナミクスとエンベロープのパラメータが設定できる。また演奏中にこれらのパラメータをリアルタイムで制御できる。

新 UPIC システムは商用システムとして提供することになっている。最後に新 UPIC システムの仕様を記す。

1. ホスト計算機として、IBM PC-AT386(最小 4MB メモリ、ハードディスク、マウス、MIDI

ボード)、オプションとしてディジタイザタブレット (A0 から A4)

2. リアルタイムシンセシスユニットとして、
   - FM 方式の加算シンセシス
   - 64 個の発振器 (44.1kHz で内挿演算可)
   - 変換ボード:4 出力 /2 入力オーディオチャネル、 AES/EBU ボード (最大 4 枚まで可)
3. OS として、マイクロソフト WINDOWS3.x を使用
4. リアルタイムコントロールとして、
   - ページコントロール
     テンポ、ページ切り換え、位置決め、オーディオチャネルごとのダイナミクスとエンベロープ制御
   - シークェンスコントロール
     テンポ、位置決め、シークェンス切り換え
   - グループコントロール
     ソロ、ミュート、周波数変調、出力チャネル設定、波形設定 (最大 64)、周波数テーブル設定 (最大 4)、エンベロープ設定 (最大 128)、振幅テーブル設定 (最大 4) 等
   - 演奏中の編集機能
     ページの演奏中にユーザが波形、エンベロープ、変換テーブルを変えることができる機能

4.2.2 MAX (平野)

概説

MAX は、当初 IRCAM の 4X シンセサイザーをコントロールするために、 Miller Puckette によって、作られたテキストベースの言語であった。その後、 Macintosh で MIDI を扱うためのグラフィックな開発環境としてインプリメントされた。現在のバージョンは、 1989 年以来、 David Zicarelli によって改良され続けてきたものである。

MAX は、一般の MIDI シーケンサーやボイシングソフトに限界を感じている人のために作られた、 MIDI アプリケーションの開発環境である。実際、 MAX によって、 MIDI 機器を自由にコントロールすることが出来る。例えば、自動作曲、伴奏、シンセサイザーの音色編集などのアプリケーションを MAX 上で、作成することができる。おおよそ、 MIDI とコンピュータを使ってできる大抵のことが MAX 上でプログラム可能である。 MAX は、 Macintosh のアイコンベースのユーザーインターフェイスとグラフィック能力を十分活用した、優れたグラフィカルプログラム言語 (graphical programming language) であり、プログラムは、グラフィックに表示されたオブジェクト (graphical objects) を線で結ぶことによって、おこなわれる。また、 MAX 上で作成された、プログラムはリアルタイムに動作し、また動作中にプログラムの編集が可能である。 MAX 自体は、 C 言語によって記述されていて、 C 言語の経験者は、自分で書いた C のプログラムと MAX を組み合わせることや、 HyperCard の XCMD のように、 MAX のオブジェクトを C 言語で記述することもできる。

MAX の構成要素

MAX は、オブジェクト指向の MIDI アプリケーション開発環境で、グラフイックに表現されたオブジェクトの入力と出力を接続することでプログラムを行なう。 MAX にはあらかじめ、 MIDI を扱うためのオブジェクトとスライダー、トグルスイッチなどのグラフィックユーザーインターフェイス用のオブジェクトが多数用意されている。そのため、比較的簡単に優れたグラフィッ

クユーザーインターフェースを持つMIDIアプリケーションを作成することができる。ここでは、MAXのユーザーズマニュアル[PZ90]を基にしてMAXの基本的な構成要素を説明する。

ウインドウ　MAXのプログラムは、パッチャー(patcher)と呼ばれるパッチャーウィンドウ(patcher window)の上で作成される(図4.7参照)。

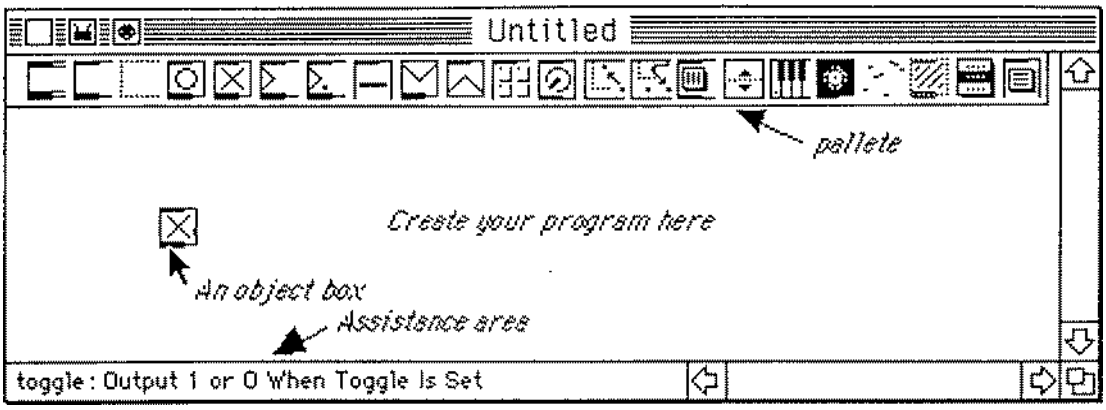


図 4.7: パッチャーウィンドウ

パッチャーウィンドウは、グラフィックソフトのように動作し、ウィンドウ上部のパレット(palette)よりオブジェクト(object)を選び、ウィンドウ上の好きな位置にオブジェクトを置くことができる。そして、複数のオブジェクトの入出力を線で結ぶことによりプログラムを作成していく。また、ウインドウ左上部に2つのアイコンがある。最左端のアイコンは、クリックするたびにパッチャーウインドウの状態がロック(lock)、アンロック(unlock)と切り替わる。パッチャーウインドウがロックされていると、ウインドウ上部のオブジェクトパレット(object palette)が消え、プログラムエリア上のユーザーインターフェイス用のオブジェクトをクリックすることで、ユーザーインターフェイスとして操作できる。この状態をrun modeと言う。一方、MAXのプログラムを作成する場合は、unlockの状態にして行なうが、これをedit modeという。その隣のMIDI conectorの形をしたアイコンは、そのパッチャーウインドウのMIDIの入出力を行なうかどうかを切り替えるスイッチの役目を持っている。また、いくつかのオブジェクトのテキストデータを書いたり、編集したりするテキストウィンドウ(text window)や、テーブルオブジェクト(table object)の数値の配列データをグラフィック化して見たり、編集したりするのに使用されるテーブルウィンドウ(table window)がある(図4.8参照)。

マックスウィンドウ(MAX window)は、ロードされた外部オブジェクト(external object)や、プログラミングのエラーメッセージなど、MAXからユーザーへのメッセージを表示するウィンドウであり、MAXを起動したとき最初に表示される(図4.9参照)。

オブジェクト　ここでは、今までの説明の中に登場したオブジェクトについて見ていくことにする。オブジェクトの多くは、そのオブジェクトを表すボックスの上部に入力(inlet)、下部に出力(outlet)を持つ。あるオブジェクトのoutletから他のオブジェクトのinletにドラッグすることにより二つのオブジェクトをパッチコード(patch cord)で接続することができる。最も一般的に使われるオブジェクトのタイプは、オブジェクト・ボックス(object box)と呼ばれ、上下にそれぞれ二本の線を持つボックスによって表される(図4.10参照)。

オブジェクトボックスの実際の動作は、ボックスの中に書かれる単語(例えば、“delay”のような動詞)や記号(“+”, “*”, “>”等の演算子)によって異なる。典型的なオブジェク

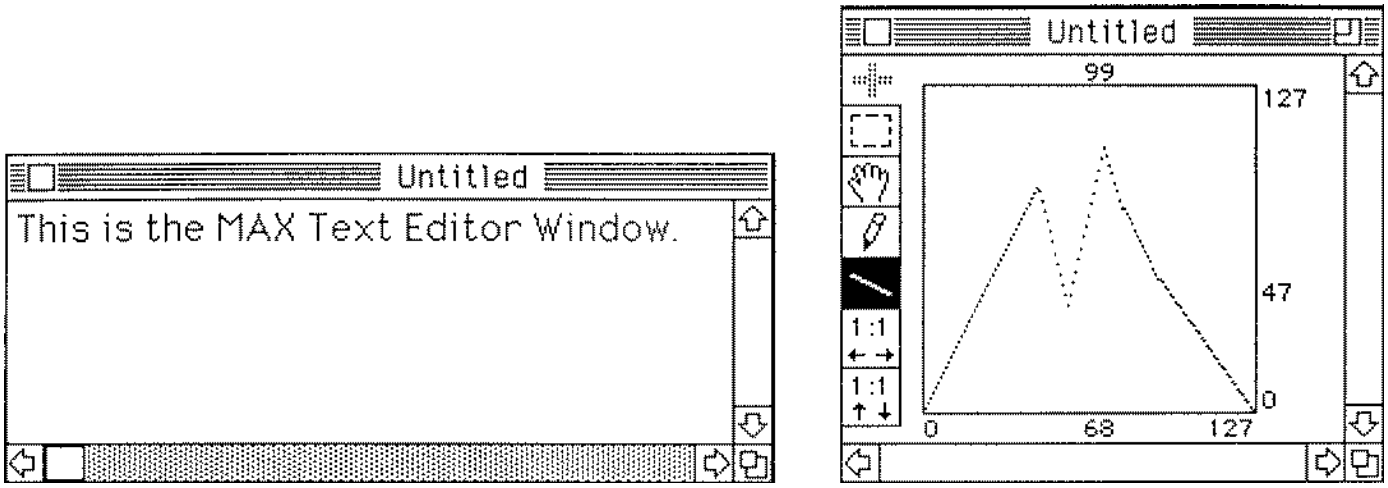


図 4.8: テキストウィンドゥとテーブルウィンドウ

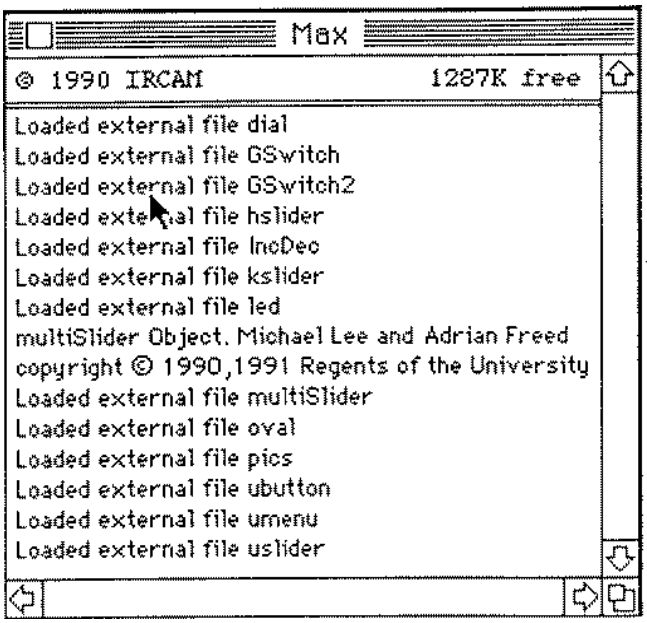


図 4.9: マックスウィンドウ

トは inlet から入力された情報に応じて、そのオブジェクトで処理した情報を outlet から出力する。オブジェクトの inlet、outlet はそれぞれ異なる目的を持っており、その目的は、マニュアルに記述されている他、パッチャーウィンドウ下部の assistance area に簡潔に表示される。この表示は、自分の知りたい inlet、outlet、の上にマウスを持っていくだけで自動的に表示される。また、オブジェクトをセレクトした後に “ commannd ”+“ h ” key を押すことにより、そのオブジェクトを使用したサンプルプログラムのパッチャーウィンドウが開くので、これを利用してオブジェクトの機能を知ることができる。いくつかのオブジェクトは、inlet 又は、outlet を持っていない。なぜならば、それらのオブジェクトは、それを含んでいるパッチ以外（他のパッチ、MIDI 等）とデータの送受信を行うためである。また、ユーザーインターフェース・オブジェクト (user interface object) がある。ボタン、ダイアル、スライダー等の形をしているこれらのオ

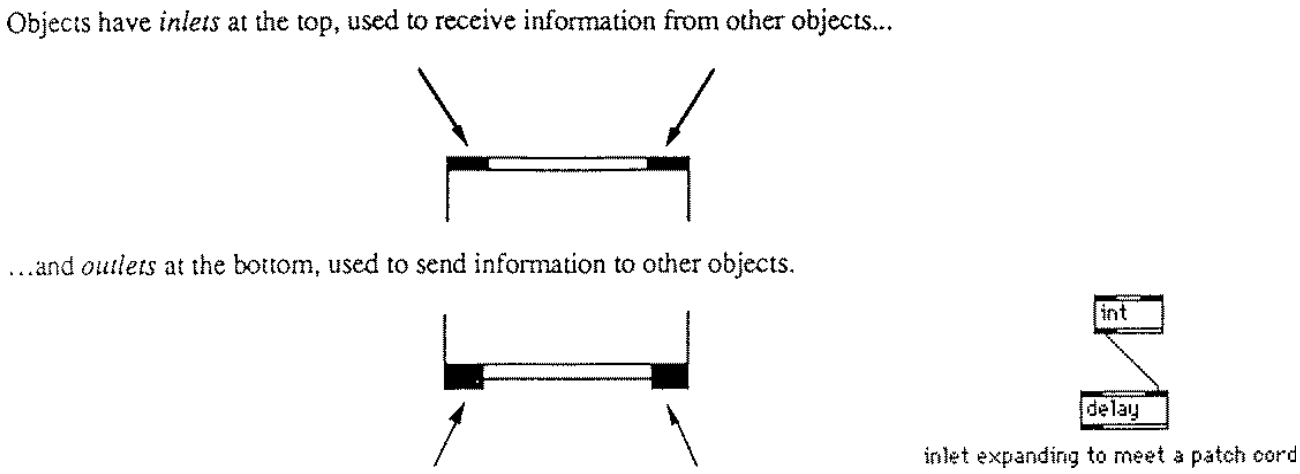


図 4.10: inlets, outlets & patch cord

ブジェクトは、他のオブジェクトからのデータを受信し表示するだけでなく、run mode において、オブジェクトをマウスでクリックしたりドラッグしたりして、データを送信することができる (図 4.11参照)。

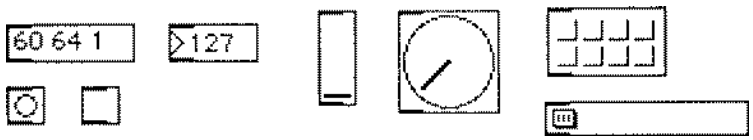


図 4.11: ユーザーインターフェースオブジェクト

その他に、プログラム上にコメントを残すだけで、何の動作もしない、コメント・オブジェクト (comment object) がある。オブジェクトの中には、オブジェクト名 (object name) の後ろに、単語や数字を付け加えるものがある。付け加えられる単語は、アーギュメント (argument) と呼ばれ、オブジェクトによって必ず要求される場合と、付加的な場合とがある (図 4.12参照)。

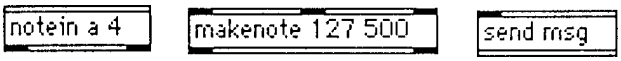


図 4.12: アーギュメント付きオブジェクト

アーギュメントが必須のオブジェクトにアーギュメントを付け忘れた場合、MAX は、MAX window に error message を出力し、そのオブジェクトは作成を拒否される。付加的なアーギュメントは、主にそのオブジェクトの初期値的な意味を持つ場合が多く、inlet からデータが与えられるまで、そのオブジェクトはアーギュメントのデータで動作することになる。現在、MAX にはおよそ 100 個のオブジェクトが用意されているが、バージョンアップする度にその数は増加している。

数値 / メッセージ　MAX のパッチコードを通じて送受信されるデータは Number と Message に分けられる。以下それぞれについて説明する。

- **Numbers**
  Number は、整数型 (int) と小数点型 (float) に分けられる。MAX において小数点型から整数型に変換される場合、少数部分は切り捨てられる。例えば、4.1 も 4.9999999 も 4 に変換される。
- **Message**
  ここでいうメッセージは、あるオブジェクトから他のオブジェクトに受け渡される情報を意味する。メッセージは、数値、スペースで区切られた数値の配列、単語 (MAX では symbol と呼ばれている) 、後に数値や他の単語が続く単語がある。メッセージの最初には、セレクター (selector) というメッセージの種類を表すものがくる。そしてセレクターの後に来る数値やシンボルは、アーギュメントと呼ばれる。

以下に MAX のメッセージで使われるセレクターをまとめてみる。

- **int**
  整数値 (integer number) のみのメッセージ。この場合セレクターは、MAX によって自動的に int として認識されるので、特に " int 127 " のように int を付ける必要はない。
- **float**
  一個の少数値のみのメッセージ。MAX は、小数点によって自動的に float と認識するので、int のときと同様に float を数値の前に付ける必要はない。
- **list**
  二個以上の空白 (space) で区切られた数値からなるメッセージ。MAX は、その後に続くものがある数値は list と認識するため、list も付ける必要がない。また初めが数値であれば、その後にはオーギュメントとしてシンボルを含めることもできる。
- **bang**
  このメッセージは、MAX に固有なもので、bang を入力されるオブジェクトの行う処理の開始又は再開を指示するトリガー的な役目を持つ。例えば、" random " というオブジェクトでは、bang を受け取るたびにランダムな数値を outlet から出力する。
- **symbol**
  多くのシンボルは、ある特定のオブジェクトに対してのみ意味を持つ。例えば MIDI の演奏情報を記録、再生するための "seq" というオブジェクトは " start ", " stop ", " record " というシンボルに応答して動作するが、" * " (multiply) というようなオブジェクトに対しては、無意味である。オブジェクトは、それ自身が理解できないメッセージを受け取った場合、エラーメッセージを MAX window に出力する。

メッセージボックス (Message box) は、メッセージの送信と表示を行い、256 項目までのボックス内に書き込まれた文字、数値など何でも含むことができる。メッセージボックスは、マウスクリックされたとき、又は、inlet に何らかのメッセージが入力されたとき、それ自身のメッセージを outlet より出力する。

メッセージの順序　オブジェクトが、複数の outlet を持っていることがある。この場合、メッセージが同時に出力されることになっているが、厳密には右から左の順に出力される。また、一つの outlet から複数の patch cord が出ている場合の出力される順序もその path cord が接続されているオブジェクトの画面上の位置が右から左の順序でおこなわれる。もし 2 つのオブジェクトが垂直に並んでいた場合は、下から上の順序で出力される (図 4.13参照)。

オブジェクトが、複数の inlet を持つ場合、オブジェクトは、右から左の順にメッセージが入力されることを見込んでいる。一般にオブジェクトは、最左端の inlet にメッセージが入力されるまでメッセージはオブジェクト内に保存され、最左端へのメッセージの入力をトリガーにして、

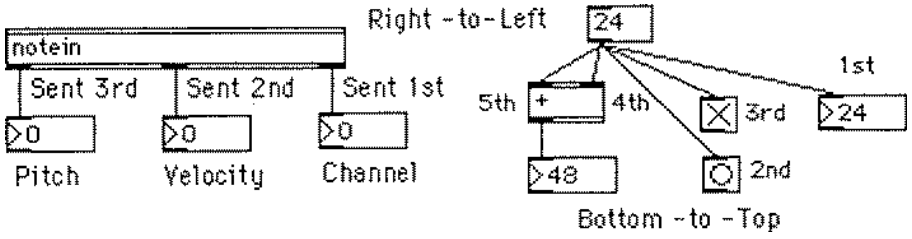


図 4.13: メッセージの順序

表 4.1: MIDI 送受信オブジェクト

Type of MIDI Message	Trasmitt Object	Recieve Object
Key ON/OFF	noteout	notein
Pitch Bend	bendin	bendout
Control Change	ctlin	ctlout
Polyphonic Key Pressure	polyin	polyout
After Touch	touchin	touchout
Program Change	pgmin	pgmout
Real Time	rtin	
System Exclusive	sysexin	

演算等の動作を実行する。また、多くのオブジェクトは、数値のリストを左の inlet から受け取ることが出来る。この場合、異なる inlet に右から左の順に数値が入力されたのと同じ結果となる。

MIDI オブジェクト　ここまで、MAX というプログラミング環境自体がどのようなものであるか見てきたので、ここで少し、MAX の特徴でもある、MIDI 用のオブジェクトについて、見ていくことにする。MIDI 用のオブジェクトは、MIDI の送受信をシリアルポートと直接又は MIDI Manager、OMS(OpcodeMIDI System) を通じて行なうオブジェクトと MIDI のデータを処理するオブジェクトに大別できる。MIDI の送受信を行なうオブジェクトに共通して言えるのは、MIDI を受信するオブジェクトは inlet を持たず、送信するオブジェクトは outlet を持たないことである。なぜならば、これらのオブジェクトは、パッチの “外の世界” とメッセージのやりとりを行なうからである MIDI 送受信用のオブジェクトとして、midiout、midiin がある。この 2 つは、MIDI データそのものを送受信する。これに対して、ある特定の MIDI データを受信したり、送信したりするオブジェクトに以下のようなものがある (表 4.1参照)。

これらのオブジェクトは、名前からも類推されるようそれぞれが、MIDI のステータスと結びついていて、関連するステータスの MIDI メッセージの送受信のみを行なう。次に、MIDI データの処理、操作等を行なうオブジェクトを見ていくことにする。midiparse は、midiin object から受け取った数値を MIDI メッセージのタイプによって分類する。また seq object は、簡単な MIDI シーケンサーの機能を持ち、MIDI メッセージの記録と、再生速度を可変しての再生が可能である。mtr object は、マルチトラックの MIDI シーケンサーで、各トラック個別の記録、再生が可能である。この他にも、MIDI 用のオブジェクトには、様々なものが用意されているが、後はオブジェクト名を列記するだけにとどめる。

- **Note ON/OFF** に関連するオブジェクト
  follow, makenote, midiformat, poly, stripnote, sustain,xnotein,xnoteout
- **Patch Editor** に関するオブジェクト
  lib, libbang, libfrom, libto, libnames, libwitch, libtofrom
- その他の **MIDI** オブジェクト
  sxformat,Borax,xbendin,xbendout, etc.

プログラム例

ここでは、実際に MAX のプログラムの例をあげて説明を行なう。図 4.14は、MIDI DELAY & TRANSPOSER とも言うべきプログラムで、入力された MIDI の Note ON/OFF の信号を一定時間遅らせて、さらに完全 4 度 (5 semitones) 上げて出力する。

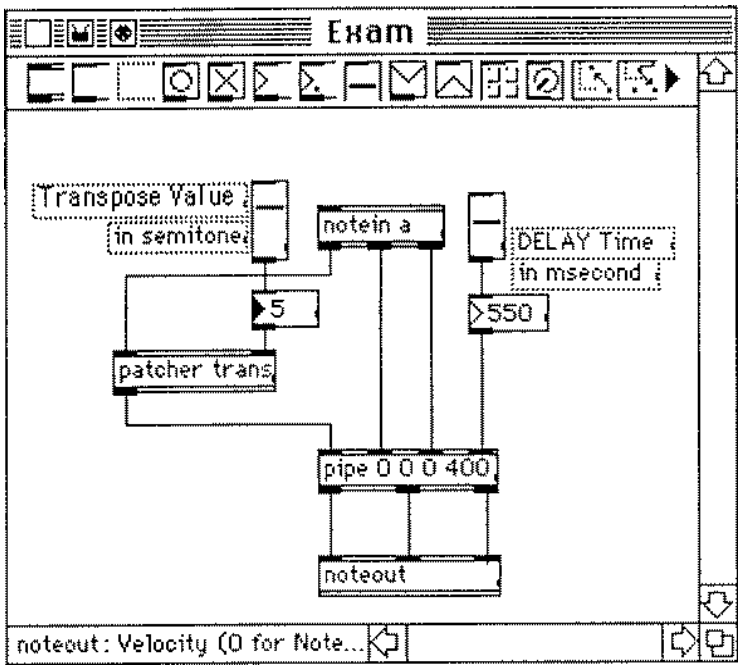


図 4.14: MIDI DELAY & TRANSPOSER パッチャー

“ notein ” オブジェクトの 3 個の outlet から、MIDI の Note ON/OFF メッセージが、MIDI channel、velocity、pitch、に分けて出力される。ここで “ notein ” オブジェクトの “ a ” というアーギュメントは、Macintosh の Modem port から入力された MIDI 信号を選択していることを示す。“ pipe ” オブジェクトは、右端の inlet から入力された時間 (1/1000 秒単位) だけ残りの inlet から入力されたデータを遅らせて出力するオブジェクトである。このオブジェクトの 4 個のアーギュメント “ 0 0 0 400 ” は、左から順に 4 個の inlet の初期値を与えている。“ noteout ” オブジェクトは、“ notein ” オブジェクトと逆に、3 個の inlet から入力されたデータを MIDI の Note ON/OFF メッセージとして出力する。“ notein ” オブジェクトの両側にある、オブジェクトは、“ slider ” オブジェクトと呼ばれる、ユーザーインターフェイス用オブジェクトの一種である。レバー部分をマウスでドラッグすることにより、それぞれトランスポーズの量、デイレイタイムを設定することができる。また、それぞれの下についている、メッセージボックスは、接続されている “ slider ” オブジェクトの値を表示している。“ patcher ” オブジェクトは、他のパッチャープログラムをサブプログラムとして取り込むことができる。アーギュメントの “

trans ” の部分でサブパッチャー (Sub patcher) 名を指定している。サブパッチャー “ trans ” では、2 つの入力の和を出力するようになっている (図 4.15参照)。

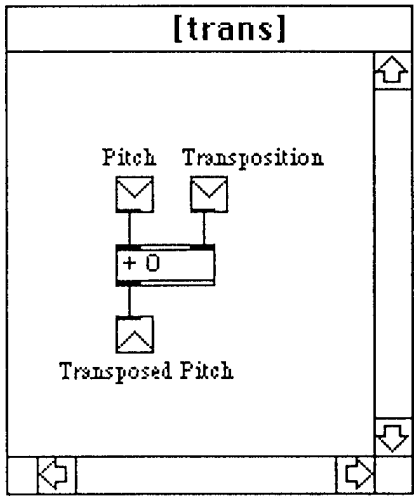


図 4.15: サブパッチャー

“ patcher ” オブジェクトの inlet 、outlet の機能と個数はサブパッチャーの “ inlet ” オブジェクト “ outlet ” オブジェクトの機能、個数と対応している。(サブパッチャーをサブルーチンに、inlet と outlet のデータを、引数と戻り値に置き換えて考えると解かり易いかもしれない。) さて、次に MIDI Auto Arpeggio を考えてみる。ここでは、入力された MIDI Note ON/OFF を元に、そのピッチの何度か異なる Note ON/OFF のデータを決められた時間間隔で出力する機能を想定する。例えば、C4 の pitch のデータが入力されたら、1/4 秒 (250msec) 間隔で、E4、G4 が出力される。ここでは、先程の MIDI DELAY & TRANSPOSER パッチャーを複数利用する方法を採用してみた (図 4.16参照)。

このような、同じプログラムを複数回利用する場合、多くのプログラム言語ではループを形成する方法を取ってプログラムする方が簡単にできる。しかし、MAX では、同じパッチャープログラムを複数、同時に走らせることができるため、こういった例を挙げてみた。この手法は、一つのパッチャーが、あるひとつのパートを演奏する役割を持っている場合。合奏を行なわせるには、パートの数だけのパッチャーを走らせることで、非常に簡単に実現できることを示唆している。ここに、示した 2 つの “ EXAM ” というプログラムは、上部のものが、run mode で下部のものが edit mode になっている。プログラムの実行時には、ユーザーインターフェイス用のオブジェクト以外は、あまり必要でない場合が多い。そのため、MAX には run mode 時には、見えなくなるオブジェクトを指定することができる。上部のパッチはその一例である。さらに、パッチャーウインドウ上に、他のペイントソフトを使って作成した、グラフィックを重ね合わせることが可能である。これらによって、オリジナルなコントロールパネルをデザインすることが可能となる。

これからの MAX

MAX は、はじめに述べたとおり、MIDI プログラムの開発環境として生まれた。しかし、現在は MIDI のみではなく、さまざまな発展、改良が行なわれている。例えば、Macintosh シリアルポート等を利用して映像機器のコントロール、Macintosh 用の DSP ボードのコントロール等に使用されている。実際に、MIDI 以外の外部機器 (LaserDisk、CD-ROM) を制御するオブ

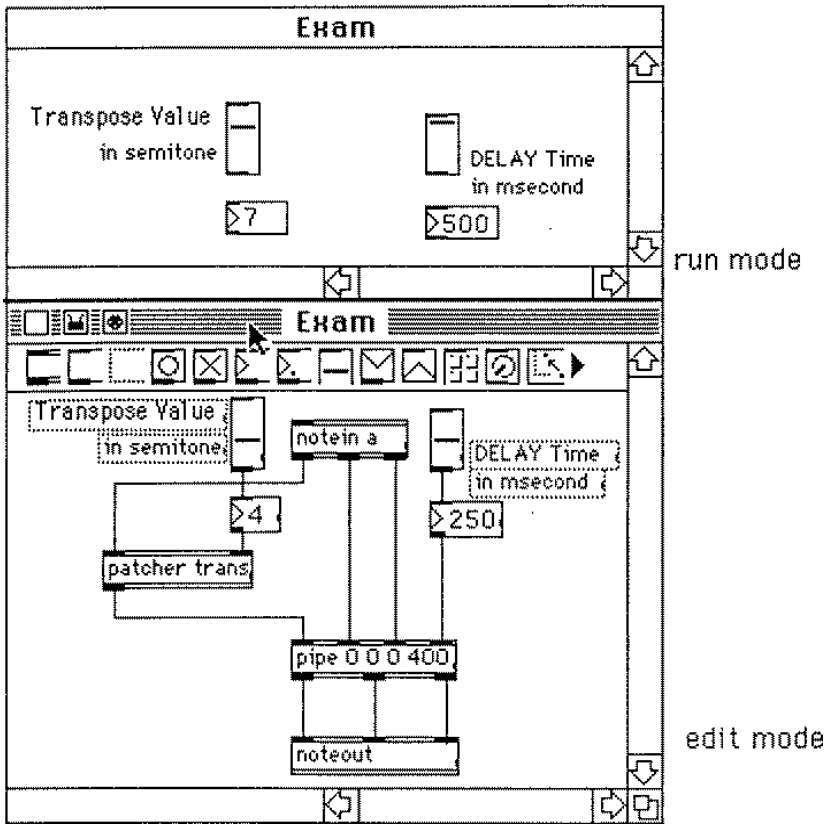


図 4.16: MIDI Auto Arpeggio パッチャー

ジェクトが既にいくつか見受けられる ([FWZ91] 参照)。その他、ニューラルネットの機能を持ったオブジェクトの開発 ([LFW91] 参照) がある。トランスピュータを搭載した Macintosh 用の拡張ボードを使用して MAX の動作を高速化させる、トランスピュータ版 MAX とでも言うべき開発環境に、T-MAX がある ([PL91] 参照)。

実際、MAX の優れたグラフィカルユーザーインターフェイスは、計算機音楽の開発環境の手本として模倣されるほどである。特に欧米においては、大学の計算機音楽の授業で教材として取り上げられたり、MAX を使用した作品のコンサートが開催されるなど、かなりポピュラーになっている。一方、日本ではマニュアルが日本語に翻訳されていないこともあり、まだマイナーな存在である。当然ながら MAX にもいくつかの問題点が存在する。その中で強調したいのは、動作環境とコンパイラーの問題である。MAX で作られたプログラムは、現在 MAX 上または MAX-Play という MAX によって作られたプログラムを動作させるためだけのソフトがなければ、実行させることができない。つまり、MAX または MAX-Play を所持している人しか MAX によって作成されたプログラムを使用できないことになる。ここで、MAX のコンパイラが必要となるのである。コンパイルすることにより、Macintosh の OS 上からプログラムの実行が可能になり、さらにプログラムサイズの縮小、動作速度のアップが可能となるであろうことは想像に難くない。MAX は、これからもバージョンアップするに従い様々なオブジェクトが追加され、さらに改良されていくであろう。できれば、コンパイラなどの基本的な部分でのさらなる開発、改良が望まれる。

4.2.3 ISPW (平野)

概説

IRCAM Musical Workstation（以下 ISPW[1]）は、NeXT computer をホストマシンとし、これに Intel i860 RISC Processor を 2 個、Motolora DSP56001 を 1 個搭載した、信号処理ボード（以下 i860 ボード）から構成されるシステムで、IRCAM に於て実時間の信号処理とイベント処理を容易に実験できるようにデザインされている。ISPW の狙いは、インタラクテイブな作曲 (interactive music composition) とパフォーマンスアルゴリズム (performance algorithms) である。実際 ISPW を使用して何ができるのであろうか？簡単に言い切ってしまえば、デスクトップで自分のオリジナルのデジタルシンセサイザーが作れるのである。前章で紹介した MAX は、MIDI Message を自由に扱うことができる環境を私達に提供してくれたわけだが、ISPW に於ては、デジタルオーデイオ信号が実時間で扱えるようになる。実際このデジタル信号処理のために、i860 ボードが必要とされた。MAX のときと同様に信号処理用に準備されたオシレーターやフィルター等のオブジェクトをパッチコードで接続していくだけで、昔のアナログのモジュール式シンセサイザーの様に自由な音作りがデジタル領域で可能となる。さらに、NeXT に接続されているハードデイスクにマルチトラックデジタルレコーデイングも可能なので、1970 年代の電子音楽スタジオを NeXT のデスクトップ上で実現できるシステムと言えよう。当然のことかもしれないが、ホストコンピュータである NeXT 上で作られた、自動作編曲プログラムやパフォーマンスアルゴリズム、などの出力によって IRCAM ボード上に作成された、信号処理プログラムをドライブすることも可能である。

ハードウエア

ここでは、ISPW のハードウエア構成について説明する。イントロダクションでも述べたように、ISPW のハードウエアは、NeXT ホストコンピュータと i860 ボードと呼ばれる、信号処理ボードによって構成されている。NeXT 一台につき最大 3 枚の i860 ボードが NeXT Bus に接続可能である。NeXT Bus は、MIT/Texas Instruments NuBus を拡張したもので、32bit のデータを 40nsec で転送可能である。このことより、理論的には NeXT Bus に接続されたボード間で最高 100MBytes / sec のデータ転送が可能である。しかし、実際にはその半分以下の速度になってしまう。NeXT に i860 ボードを 3 枚搭載した ISPW の場合、最大 3 台の同じ ISPW と相互に接続可能である。この場合、図 4.17に示すように、ある NeXT の i860 ボードと他の NeXT の i860 ボードに接続する。つまり、3 枚の i860 ボードを搭載した NeXT は、他の 3 台の i860 ボードと接続できる。こうやって、計 4 台の NeXT を使用し最大 24 個の i860 を持つシステムを構築することが、可能である。

次に、i860 ボード自体の構造について話を進めていく。図 4.18からも分かるように i860 ボードには、2 個の Intel i860 RISC プロセッサーと 1 個の Motolora DSP56001 が搭載されている。i860 プロセッサーはそれぞれ 8MBytes 単位で最大 32 MBytes までのダイナミック RAM を接続することができる。

そして、プロセッサー間の通信はそれぞれのプロセッサーが持つ RAM に読み書きすることで行なわれる。同じボード上のプロセッサー間の通信は、クロスバースイッチ (crossbar switch) を介して行なわれる。クロスバースイッチは、2 つのブロックから成り、それぞれが 4 個の 32bit のポートを持つ。一方のブロックは 2 つの i860 プロセッサーの、64bits のデータバスの内 LSB (Less Significant Bit) 32bits を受け持ち、もう一方は MSB (Most Significant Bit) 32bits を受

[1]参考文献のなかでは、IMW(IRCAM Music Workstation) と略されているが、現在は、ISPW(IRCAM Signal Processing Workstation) という略称がより一般的になっているので、ここでは ISPW という略称を使用する。

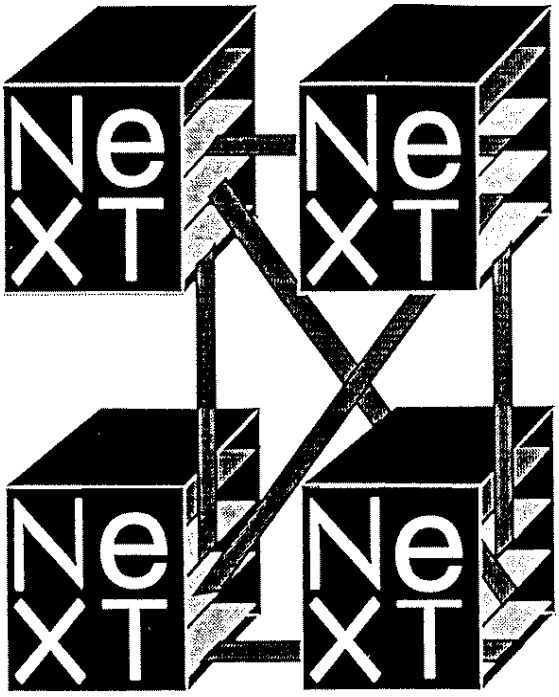


図 4.17: NeXT & i860 ボード相互接続図

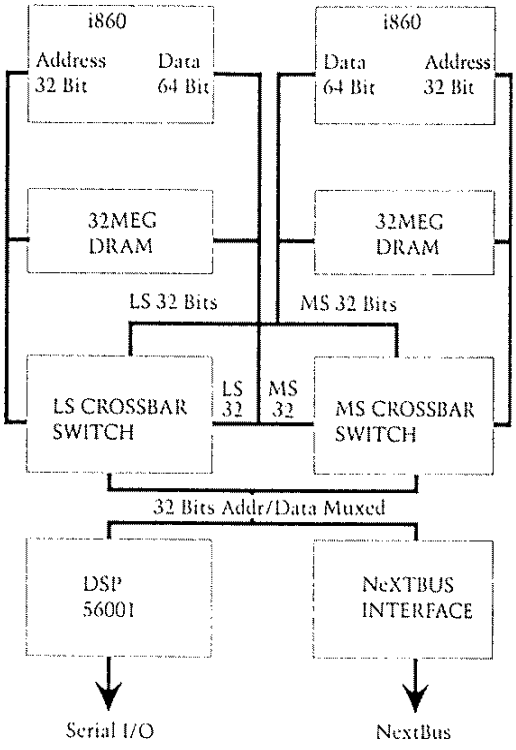


図 4.18: i860 ボード

け持っている。つまりクロスバースイッチの 4 個のポートのうち 2 つは、i860 プロセッサーのデータバスと接続されていることになる。残りのうちの一つは、それぞれの i860 プロセッサーのアドレスバスと接続され残りの一つは、MBUS (32-bit address/data multiplexed bus) と接

続されている。このMBUSは、I/O及びボード外との通信に使用される。そして、このMBUSは、I/Oプロセッサーとして使用されているDSP56001プロセッサーと接続されている。DSP56001は、44.1kHzサンプリング周波数の双方向のシリアルデジタルオーデイオを4チャンネル及びMIDIやその他のコントローラのための、RS-422シリアルポートのI/Oとしての働きを持つ。また、i860プロセッサーのローカルメモリーや、異なるボードのi860プロセッサーのメモリー間でのデータ転送の際のDMAとしての働きを持つ。実際、i860ボードの信号処理ボードとして見た時のパフォーマンスはというと、サンプリング周波数が44.1kHzのデジタルオーデイオ信号の場合、一枚のボードで500の補間なしのテーブルルックアップ式のオシレータ、500個の2次のフィルター、1300の基本的なベクトル演算、1024点の複素型FFTを1msecで（これのみ1個のi860プロセッサーあたり）行なう能力がある。

ソフトウエア

図4.19に、ISPWのソフトウェアの構成を示す。中央の太線より下の部分は、i860プロセッサー上で走るプログラムを表している。上の部分は、ホストコンピュータ及び、マルチプロセッサーとの通信を表している。ここでは、図4.19の中よりISPWの特徴的なソフトウエアである、CPOS、FTS、Animal、MAX、Signal Editor、Recorderについて、簡単に説明していくことにする。

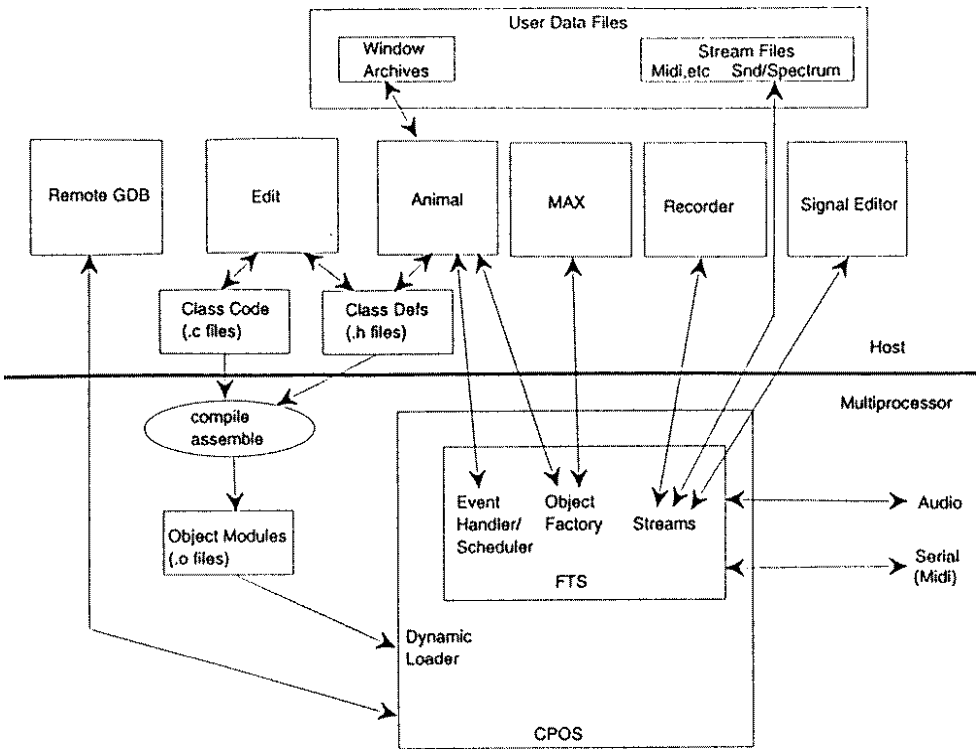


図 4.19: ソフトウェア構成

- **CPOS（CoProcessor Operating System）**
  CPOSは、複数のi860プロセッサーの上で走るオペレーテイングシステムである。CPOSは、リアルタイムのイベント処理と信号処理を効率的に行なうためのカーネルのパラメータを制御している。実際は、プロセスマネージメント、メモリーマネージメント、プロセ

ス間通信、ファイルコントロールを担っている。また、UNIX SystemV との部分的な互換性があり、現在 40 程の UNIX と共通のシステムコールを持っている。

- **FTS(Faster Than Sound Toolbox)**
  FTS は、CPOS の上位レベルで走り、複数のプロセッサーにまたがる、信号処理のスケジューリングや、メモリーマッピング、ミュージックイベントの処理をサポートしている。また後で述べる、MAX や Animal が複数のプロセッサーを意識せずに使用できるのも FTS の役割による。
- **MAX**
  イントロダクションでも述べたように、MIDI だけでなく、リアルタイムの信号処理も可能な MAX がインプリメントされている。MIDI アプリケーションの開発環境としての MAX の説明は、前章で述べたので、ここでは信号処理のオブジェクトについて多少コメントする。現在は、24 個の信号処理用のオブジェクト（tilde object）が用意されている。tilde object は、全てオブジェクト名の最後に“~”が付いていて、イベント処理用のオブジェクトと区別されている。tidle object の例をいくつか挙げることで説明に換えることにする。

  signal~ イベント処理用のデータを信号処理用のデータ形式に変換する。
  line~ エンベロープジェネレータの作成に使用される。
  osc~ テーブル参照型の余弦波の発信器。
  ***~** デジタルオーデイオ信号の積算。
  dac~ D/A 変換を行なうオブジェクト。これによって、MAX の信号処理の結果を音として聞くことができる。

- Animal (Animated Language)
  Animal は、実時間信号処理やイベント処理の実験ソフトを作成するために、設計されたプロトタイピングツールである。Animal は、計算機音楽のアプリケーション作成に適したオブジェクト指向型のグラフィック言語であり、音楽データのリスト編集、シンセサイザーの音色エデイット、サンプリングシンセの操作インターフェイスソフト等の作成に適した、グラフィックアプリケーションを作るためのツールソフトである。
- Signal Editor
  Signal Editor は、デジタルオーデイオ信号の時間及び周波数領域での表示と編集を行なうためのツールである。周波数領域でのツールは、“ SpecDraw ”と呼ばれ、デジタルフィルターの時間的変化の様子が、ソナグラフのように、直交する時間軸—周波数軸上の平面に、そのレベルをグレースケールで表現されて描かれている。この時間—周波数領域に、多角形を描くことで容易にダイナミックデジタルフィルターを構成することができる。
- Recorder
  “ Universal Recorder ”と呼ばれるアプリケーションは、イベント及びオーデイオ信号をリアルタイムにメモリーやデイスクに記録するためマルチトラックレコーダーを模範にしたユーザーインタフェースを持っている。レコーダーは、時間的に正確な再生のため、FTS からのオーデイオ信号やメッセージを使用している。

まとめ

ISPW の概要を、ソフトウエアとハードウエアの両面から見てきた。イントロダクションでも少し述べたが、電子音楽スタジオがデスクトップで実現できるマシンである。それも、ソフトウエアにより自由にその機能を構成することができる。今までも、ワークステーションや DSP(Digital Signal Processor) の組み合わせで可能ではあった。しかし、それを使いこなすためには、音楽的な能力の他に、高度な LSI 技術、信号処理技術、プログラミング能力が必要とされた。しかし、

ISPW においては、CPOS、FTS などのリアルタイム処理に適した基本ソフトと MAX、Animal といったユーザーインターフェイスに優れた開発環境によって、それほど専門知識のない音楽家や、研究者にも簡単にオリジナルなプログラムを開発できる点は、非常に評価できる。そのため ISPW は、これからの計算機音楽の研究、制作の場において、さらに普及していくと考えられる。

4.2.4 その他のシステム (鈴木)

前節まで、主として音楽家や研究者を対象とした音楽制作ツールを紹介して来た。ここ数年の最新の傾向として、計算機を用いた音楽制作ツールは、これ以外にも大きな広がりを見せている。これには、大きく 2 つの流れがあると考えられる。

(a) パソコンベースのシステム: Macintosh 等のパソコンをベースにした低価格システム
(b) プロ用の大規模システム: 商業音楽制作に用いられる専用高機能システム

以下、上記 2 点について簡単に解説を行い、本節のまとめとする。

(a) パソコンベースのシステム

この分野はここ数年で急速に発展してきた。音楽制作に比較的低価格なパソコンを用いて、MIDI や PCM オーディオファイルを扱い、ユーザがインタラクティブに音楽を作っていけるツールである。使用するパソコンとしては、GUI が優れている Mac(Macintosh) をベースにしているものが多く、ここではそれらを簡単に紹介する。他にも、IBM-PC や日本国内では PC9801 を用いたものがある。

Mac 系では、当初よりその誰にでも簡単に使いこなせる GUI を背景に、優れた MIDI シークェンサが商品化されて来ている。その中でも広く使われているものとして、Mark of the Unicorn 社の Performer と Opcode Systems 社の Vision がある。さらに、MIDI 楽器だけではなく、アコースティックな音も扱いたいという要求が出て来た。ディジタルオーディオや AD 変換した PCM ファイルを、直接ディスクに、リアルタイムで録音再生できるインターフェースを Mac に付加したものが、Digidesign 社の Sound Tools である。(一般にディスクレコーディングと呼ばれる。) これによって、Mac 上で、MIDI と PCM が扱えるようになったが、音楽制作上では、MIDI 楽器と生の音を同時に扱えれば、より汎用性が得られるということから、MIDI シークェンサにディスクレコーディングを取り入れた商品が開発されている。前述のシークェンサに、2 から 4 トラックのディスクレコーディングを取り入れたものとして、Digital Performer と Studio Vision がある。

これらの、ツールの特徴としては、比較的低価格なために、アマチュアレベルの購買層を新たに開拓して、新しいジャンルを作りだしつつあるということがある。また近年では、商業音楽制作に携わるプロの音楽家やエンジニアにも、その使い易さと音の質も向上してきた (通常の DAT と同程度) ため、広く使われだしている。

計算機技術から見ると、この分野は MIDI を扱うリアルタイムプログラミングや PCM ファイルをディスクからリアルタイムで転送するためのバッファ管理技術やファイルシステムの最適化等を考える必要がある。また、MAX で説明したようにオブジェクト指向の考え方や、DSP を用いた信号処理、マルチメディア環境など最新の技術をも取り込もうとしており、現在もっともホットな話題を提供している分野とも言え、今後の発展が期待できるであろう。

(b) プロ用の大規模システム

この分野は、一般に目に触れる機会はあまりないが、プロ用音楽制作の世界のスーパーコンピュータとも言えるシステムであり、独自の発展を遂げている。

商業音楽制作においては、長い間テープを用いた制作が行われてきた。初めはアナログテープを用いていたが、それが現在ではディジタル化され、さらにマルチトラック化が進み、現在 48ch が存在する。(DAT の技術もここに入る。) しかし、テープは編集作業が困難で、現在のように音の素材をたくさん使って、何回も編集を加えながら音楽を作る作業が中心になると、もっと柔軟なシステムを使いたいという要求が出てきた。これは、計算機の世界のメディアとの対比で考えるとよくわかる。テープは大容量であるが、原理的に順次ファイルアクセスであり、ランダムアクセス (音楽では頭出し) は困難であり、自由にファイルを編集することも時間がかかり、効率的ではない。一方、ランダムアクセスが可能なディスクを用いれば、アクセスが速く、ファイルの編集も容易にできる。

音楽の世界でも、ディジタルオーディオの技術で、生の音を AD 変換器や DA 変換器を用いてディジタル信号 (PCM) として扱えるようになってきた。そのサンプリング周波数やビット幅も十分プロ仕様に耐えられるようになってきている。このようなオーディオのディジタル化と、計算機技術の進歩 (すなわちディスクの大容量化や CPU の高速化、さらには低コスト化) が結びついて、ディスクレコーディングシステム、あるいはディジタルオーディオ・ワークステーションといったシステムが開発されてきている。

これらのシステムの応用分野は、ポピュラー音楽等の商業音楽の制作はもちろんであるが、初期の段階から以下のようなシステムにおいて発展してきている。

放送局での (音声) 番組の制作、映画の画面に効果音を重ねる制作、近年では TV コマーシャルの音楽音声制作等である。これらは、ポストプロダクションと呼ばれ、映像に音楽音声を同期させる必要がある。そのため、映像の 1 フレーム (通常 30 分の 1 秒)、あるいはそれ以下に音楽音声を同期させる必要があり、テープを用いた編集では職人芸 (テープの切り貼り) が要求されていた。そのような事情から、上記のディスクレコーディングシステムは、ランダムアクセス性と編集の容易さから、真っ先にこの業界に受け入れられたと言える。

実際に開発されてきたシステムを簡単に紹介する。有名なものとしては、Fairlight と Synclavier がある。共に 10 年以上前に開発されているが、初めはいわゆるサンプラとしての楽器の使い方をしていた。サンプラは、メモリ上にレコーディングするので、容量の制約から長時間録音再生は難しい。そこでメディアとして、2 次記憶のディスクを用いようとして、前述のようなディスクレコーディングシステムが誕生した。他にも、AMS 社の AudioFile は、イギリス BBC に採用されて長く使われてきている。最近では、WAVE Frame 社の Audio Frame や DAR 社の SoundStation 等の高機能システムがある。また、スタジオ機器業界の企業である SSL や AKG、あるいはマルチトラックアナログテープレコーダを手がけていた OTARI や STUDER なども、この分野に進出している。

これらのシステムは、プロ指向であり、高価である (1000 万円以上の価格帯) が、最近では DSP を用いた信号処理までこなせるようになり、音楽制作上での総合環境としてのディジタルオーディオ・ワークステーションとしての利用が進んで行くと思われる。原理的には、音の収録以後はディジタル信号化された PCM ファイルの編集作業からミックスダウン後のマスターファイルの作成まで、全て計算機上で行うことが可能である。信号処理技術を用いれば、時間圧縮・伸張 (ピッチを変えずに時間を変える) や、ピッチ変換 (時間を変えずにピッチを変える) を行うことも可能になってきている。

しかしながら、現在ではマルチトラックの大容量の PCM データを扱う上で、解決しなければならない問題も残っている。

一つには、大容量のデータを転送する時間が長いということがある。例としては G バイトにもなるディスクのデータを他に転送したり、バックアップしたりする場合、その転送時間だけで

も何分もかかり、作業が中断してしまうということがある。この解決法の一つとして考えられるのは、可搬メディアである光磁気 (MO) ディスクの採用である。現在は、まだハードディスクに比べて性能面では劣っているが、将来の性能向上によりハードディスクに代わることが期待されている。また、大容量データを扱うデータベース技術のアプローチも有効かもしれない。

二つ目に、信号処理では、音に簡単なエフェクトをかけるような処理はリアルタイムで可能であるが、前述の時間圧縮・伸張やピッチ変換をリアルタイムかつ高品質で行うには、まだ解決しなければならない技術が残っているであろう。

最後に、より洗練されたマンマシンインターフェースの構築が重要になってきている。これらのシステムは、高機能化している反面、その操作性が必ずしもいいとは言えない。特に、計算機の専門化でない音楽家が扱うこの分野では、より一層のマンマシンインターフェースの向上が望まれている。これは、この分野だけの問題ではなく、他の計算機を用いたシステム全てについても言えることである。優れた GUI やオブジェクト指向パラダイムの採用、より人間にあった入出力デバイスの開発等、計算機科学技術のより一層の発展が期待されている。

4.3 伴奏システム（堀内）

本節ではリアルタイム・インタラクティブ・システムの一つである伴奏システム (Accompaniment System) について、その研究を概観する。伴奏システムとは人間の独奏者の演奏を聴き、その演奏にタイミングやダイナミクスを合わせてリアルタイムで伴奏を出力するシステムである。

4.3.1 伴奏システム研究の流れ

伴奏システムの研究の流れを簡単にまとめると図 4.20 のようになる。

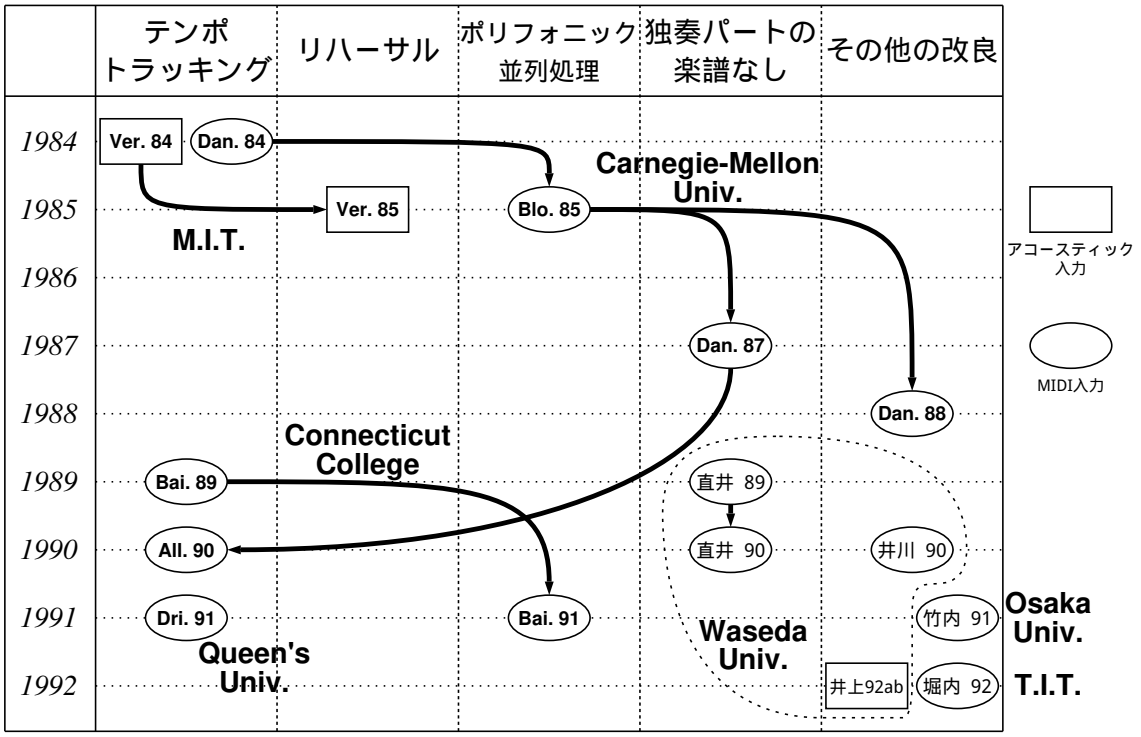


図 4.20: 伴奏システム研究の流れ

図にあるように伴奏システムの登場は 1984 年であり、そのような初期の伴奏システムの研究の中心はテンポ・トラッキング技術（特に楽譜と演奏とのマッチング）であった。その後、各システムは以下のようなテーマを中心にして、図にあるように様々な点で改良された。

- 入力情報（独奏者の演奏）と楽譜データのマッチング方法
- 音響信号からのピッチ抽出
- 演奏ミスなどに対するエラー処理
- リハーサルによる学習
- ポリフォニック入力に対する対応
- 並列処理
- あらかじめ楽譜が与えられてない演奏（即興演奏）のテンポ・トラッキング
- 装飾音、トリル、グリッサンドなどの扱い
- 音楽的規則を用いたテンポ・トラッキング
- 独奏者と伴奏者の相互作用
- 伴奏出力の制御法 etc.

以下、各研究機関ごとに個々の研究について概要を述べる。伴奏システムの研究を古くから行なっているのはカーネギー・メロン大学[2] の Roger B. Dannenberg とマサチューセッツ工科大学メディア・ラボ[3]の Barry Vercoe の二人である。そこでまず第 4.3.2 節、第 4.3.3 節で CMU と MIT とでそれぞれ行なわれている彼らの研究について図の流れに沿ってその概要を述べる。次に第 4.3.4 節でコネティカット大学[4]の Baird らの研究について述べ、第 4.3.5 節でクィーンズ大学[5]の Driesse の研究について述べる。その後は国内の伴奏システムの研究に移り、第 4.3.6 節で早稲田大学の大照研究室、第 4.3.7 節で大阪大学の井口研究室、第 4.3.8 節で東京工業大学の田中研究室で行なわれているそれぞれの研究について述べる。最後に第 4.3.9 節で伴奏システムの今後について述べる。

4.3.2 カーネギー・メロン大学での研究

CMU では R. B. Dannenberg が中心となって、伴奏システムの研究を行なっている。第 4.3.2 節では、行列計算によるテンポ・トラッキング方法を中心に最初の伴奏システム [Dan84] について述べる。その論文の翌年 Bloch らはポリフォニック入力に対するテンポ・トラッキングを行なうため、二通りのマッチング方法を提案した [BD85] 。これを第 4.3.2 節で述べる。次に [BD85] をいくつかの点で改良した論文 [DM88] を第 4.3.2 節で述べる。以上の研究では独奏パートの楽譜が与えられていたが、楽譜情報のない場合のテンポ・トラッキングについては第 4.3.2 節で述べる。ここでは独奏者の演奏する音符の時間情報とヴェロシティを用い、いくつかの規則に基づいて拍上の音符を見分けることによりテンポ・トラッキングを行なっている [DMR87] 。最後にビーム・サーチを行なうことによって、 [DMR87] のビート・トラッキングの性能を向上した [AD90] について第 4.3.2 節で述べる。

[2]Carnegie-Mellon University

[3]Media Laboratory, Massachusetts Institute of Technology

[4]Connecticut College

[5]Queen's University

リアルタイム伴奏 (Real-Time Accompaniment)

R. B. Dannenberg は 1984 年の ICMC[6] において彼の伴奏システムを発表した [Dan84] 。この論文は伴奏システムの先駆けとなり、その基礎を理解するのに大いに役立つと思われるので以下に要約する。

追従モデル

人間の伴奏者はさまざまな種類の視覚・聴覚のキューに頼っているが、コンピュータ伴奏は限られた範囲の演奏イベントしか検出できないと仮定する。ソロ演奏はイベントのストリームとして、楽譜は予測されるイベントの順序リスト (ordered list) のストリームとしてみなすことができる。コンピュータに記憶される楽譜は演奏家用の伝統的記述法ではなく、正確なソロ演奏で検出されるイベントのリストとなる。

問題は 2 つのストリームの「ベスト」マッチを見つけることである。ベスト・マッチとは 2 つのストリームの最も長い共有サブ・シーケンス (longest common subsequence) であると定義する。すなわち、できるだけたくさんの楽譜イベントを演奏にマッチさせることである。

マッチング・アルゴリズム

ベスト・マッチを見付けるために整数行列を計算する。このとき各行は楽譜のイベントに、各列は検出された演奏イベントに相当し、演奏イベントが検出されるごとに新しい列が計算される。行 r 、列 c における整数は前の列（前の演奏イベント）に対する答と現在の列の前の行から計算できる。

楽譜イベント r 、演奏イベント c までのベスト・マッチは少なくとも $r-1$ 、 c までのものと等しい。何故なら 1 つ以上の楽譜イベントはベスト・マッチを短くすることはできないからである。同様にして r 、 c までのベスト・マッチは少なくとも r 、 $c-1$ までのものと等しい。そのとき演奏イベントが一つ抜けたと考えられる。さらに楽譜イベント r が演奏イベント c とマッチするなら、マッチの長さは $r-1$ 、 $c-1$ までのものより少なくとも一つ長くなる。これらの規則は最長（ベスト）マッチの長さを計算するのに適用される。演奏イベントが検出されるごとにアルゴリズムは最長行列の列をもう一列計算する。図 4.21 に演奏イベント A G E D G B C 、楽譜イベント A G E G A B C に対する行列を示す。

		performance:						
		A	G	E	D	G	B	C
score:	A	1	1	1	1	1	1	1
	G	1	2	2	2	2	2	2
	E	1	2	3	3	3	3	3
	G	1	2	3	3	4	4	4
	A	1	2	3	3	4	4	4
	B	1	2	3	3	4	5	5
	C	1	2	3	3	4	5	6

図 4.21: ベスト・マッチの完全な計算結果。楽譜位置が報告された場所が下線で示される。

この時点でアルゴリズムはベスト・マッチの長さを教えるに過ぎず、この長さを得るためにどのイベントがマッチしたかを教えてはくれない。この情報は伴奏プロセスに必要となる。現在の演奏イベントの楽譜上の位置を知らせるアルゴリズムは、現在のイベントまでのベスト・マッチの長さを覚えていることによって達成される。それはすでに計算された行列中でもっとも長い

[6]International Computer Music Conference

値である。マッチがより長い値になる時は新しい演奏イベントが楽譜イベントにマッチしたときであり、演奏者が楽譜上の相当する位置にいることを知らせる。図 4.21 で報告されるマッチは下線が引いてあるものである。例えば D 音は演奏されたが楽譜上には無いので楽譜位置は報告しない。また B 音が演奏された時は独奏者が A 音を飛ばしたことが明らかになる。アルゴリズムは B 音に相当する楽譜上の新しい位置を正確に知らせる。

このアルゴリズムは空間と時間の両方で効率化を図ることができる。行列の中で前と現在の列しか必要とされないため、完全な行列を保存する必要はない。また実行時間を節約するために、演奏は楽譜と近いマッチであると仮定し、楽譜イベントの予測されるマッチの近くの小さな窓 (window) だけを見る。 図 4.22 は前と同じ例において演奏と楽譜に対して計算される値を

		performance:						
		A	G	E	D	G	B	C
score:	A	**1**	1					
	G	1	**2**	2				
	E	1	2	**3**	3			
	G			3	3	**4**	4	
	A				3	4	4	4
	B					4	**5**	5
	C							**6**

図 4.22: 現在の楽譜位置を中心とする小さな窓だけを用いたベスト・マッチの計算結果

示している。ただし窓の大きさは 3 である。窓の中心は以下のようにして決定される。もし前の列であるマッチが見つかり報告されたなら、そのマッチの次の行を窓の中心にする。一方、もしマッチが見つからなかったら窓を一行下に動かす。

その他のヒューリスティクス

このアルゴリズムの問題点は、余分な（間違った）イベントが検出された時に楽譜を先へ飛ばす場合である。このアルゴリズムの目的は最長マッチを生成することなので、未来のイベントにマッチしながら楽譜を先へ飛ばすことになる。これはしばしば間違った選択をしてしまう。アルゴリズムは楽譜を先へスキップすることを（妨げるのではなく）阻止するために何らかのヒューリスティクスを必要とする。そこで 3 つのヒューリスティクスがインプリメントされた。

最初のヒューリスティックは窓の中心が変化できる量を制限することである。これは結局、楽譜イベントをスキップできる比率を制限するということである。2 番目のヒューリスティックは楽譜イベントをスキップするマッチにペナルティを加えるというものである。これはマッチしたそれぞれのイベントのペアには 1 ポイントずつ加え、楽譜でスキップされたそれぞれのイベントは 1 ポイントずつ減らすことにより行なわれる。最終的に行列のある列での最高の値は、楽譜上の複数の位置の各行で取り得る。3 番目のヒューリスティックは、このような選択の余地が生じた時はいつでも楽譜の中で一番最初のイベントでマッチが起こっているとして報告するというものである。

伴奏

マッチング処理は楽譜に関するライブ演奏者の時間位置を報告する。この情報を用いて、あらかじめメモリ中に記憶されている伴奏のイベントをいつ演奏するかを決定する。

ハードウェア・クロックを参照するソフトウェア仮想時計をインプリメントする。その仮想時刻は以下のように定義される。

$$(R - R_{ref}) * S + V_{ref}$$

ここで R は実時間、R_{ref} は仮想時計が最後にセットされた（実）時間、S は仮想時計の速度、V_{ref} は仮想時計が最後にセットされた仮想時間である。仮想時計は現在の実時間を R_{ref} に割り当て、望ましい仮想時計の時間を V_{ref} に割り当てることによっていつでもセット可能である。また仮想時間の「速度」を定めるために S を任意に更新することができる。これは仮想時間が実時間に対して進む比率である。

この論文でのインプリメントでは、検出された入力イベントに応じてマッチング・プロセスがマッチを報告する時刻ごとに仮想時計がセットされる。仮想時刻はソロ楽譜の相当するイベントから決定され、実時間はハードウェア・クロックから決定される。それに加えてスピード S は仮想時計が進んでセットされる時には若干増加され、仮想時計が遅れてセットされる時には S は若干減少される。

キーボード演奏のリアルタイム・コンピュータ伴奏

J. J. Bloch らは翌 1985 年の ICMC において、上述の伴奏システム [Dan84] の問題点であったポリフォニック入力への対応を可能にした [BD85] 。ここではポリフォニック・マッチングの方法として静的マッチングと動的マッチングの二通りの方法を提案している。以下、それらを中心にこの論文を紹介する。また、この論文には [Dan84] ではあまり述べられていなかった伴奏システムの内部処理についても非常に細かく書かれているが、ここでは紙面の都合上、ポリフォニック・マッチングを中心に紹介することにする。詳しくは [BD85] を参照して欲しい。

ポリフォニック・マッチング

モノフォニック Matcher[7] に対して、ポリフォニック Matcher は、イベントが部分的にしか順序付けられていない楽譜を扱わなければならない。図 4.23 の下に示されるように、楽譜は記号の集合の連続として記述される。ここで同じ集合の記号は丸で囲まれている。ソロ演奏は図 4.23 の

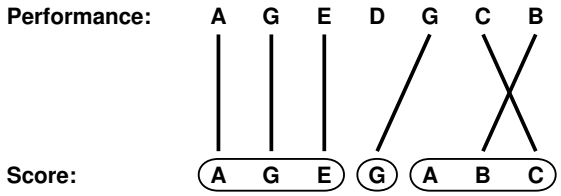


図 4.23: 演奏とポリフォニック楽譜との間の関係

上に示され、この演奏の楽譜に対するベスト・アソシエーション (best association) が線で示されている。同じ集合の中にある楽譜記号に線が引かれるとき、線はクロスしても良いということに注目して欲しい。

静的マッチング・アルゴリズム

ポリフォニック・マッチングの最も簡単なアプローチは静的グルーピング (static grouping) と呼ばれる。これは楽譜を複合音楽イベントの（全体として順序付けられた）シーケンスに分解し、ソロ演奏をこれらのイベントにリアルタイムに「解析する (parsing) 」ことを必要とする。これが達成されれば、我々のモノフォニック・マッチング・アルゴリズムを適用できる。

ポリフォニックの楽譜が分解される複合イベント (cevts) は同時に起こるすべてのノート・オン・イベントから構成される。楽譜を cevt に分解することは簡単であるが、演奏を cevt に分解

[7]Matcher とは独奏者の演奏と楽譜とを比較するモジュールで、ソロと楽譜との間の対応を Accompanist に報告する。Accompanist ではその情報に基づいて、伴奏をいつ、どのように演奏するかを決定する

することは、タイミング情報があまり正確でないため、それほど簡単ではない。ソロ演奏でノート・オン・イベントが検出されたとき、二つのノート・オン・イベントの時刻を比較することによって前の音符と同じ cevt に属するのかどうか決定できる。我々はカット・オフ・ポイントとして 100 ミリ秒を選んだ。この定数をイプシロンと名付ける。

しかしアルペジオで演奏される和音は、上の基準では誤った cevt グルーピングを行なってしまう可能性がある。しかしアルペジオの音符は次の和音の最初の音符よりも同じ和音の前の音符の方に近くなるので、それが和音に属するかどうかを調べるのは一般的に簡単である。もし 2 つのノート・オン・イベント間の時間が最初のイベントから次のイベントまでの予測される時間のある分数よりも小さかったら、2 番目の音符は最初のものと一緒にグルーピングされる。我々はこの分数の値として 1/4 を用い、これをイプシロン分数 (epsilon-fraction) と呼ぶ。

ソロ演奏の個々の音符が入力されるにつれて、即座にそれを変更不可な cevt にグルーピングする。イプシロン／イプシロン分数テストの結果によって音符が前の音符と同じ cevt にグルーピングされたり、新しい cevt にグルーピングされる。

動的マッチング・アルゴリズム

モノフォニック・アルゴリズムは記号のシーケンスをもう一つのシーケンスにマッチしたが、ダイナミック・グルーピングは記号（音符）のシーケンスを記号の集合（和音）のシーケンスにマッチする。

列が演奏記号、行が楽譜集合に対応する行列を考える。行列の各要素は $value$ と呼ばれる整数と $used$ と呼ばれる集合からなる。行 r 列 c での $value$ は楽譜集合 A_r 、演奏記号 a_c までのベスト・アソシエーションの評価関数の値である。行 r 列 c での $used$ 集合はそれに対応する $value$ に達するために、楽譜集合 r の中でマッチした記号を含む。

演奏を楽譜にマッチするため、それぞれの記号が読み込まれるたびに行列を 1 列計算する。それぞれのセルは同じ行の前のセル、前の列と前の行のセル、同じ列の前の行のセルによって計算される。図 4.24 の行列でセル w, x, y から セル z が計算される方法を述べる。 z を計算す

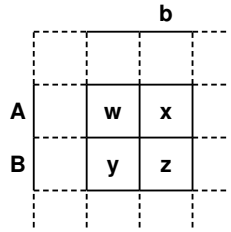


図 4.24: 4 つの行列セルに対するラベル

るアルゴリズムは以下のようになる。

```
1)  z.value := y.value;
2)  z.used := y.used;
3)  if b in B then
4)    if b not in y.used then
5)      z.value := y.value + 1;
6)      z.used := y.used union { b }
7)    endif;
8)    d := w.value + 1 - # (A - w.used);
9)    if d >= z.value then
10)     z.value := d;
11)     z.used := { b }
12)   endif
13) endif;
14) v := x.value - # (A - x.used);
15) if v >= z.value then
16)   z.value := v;
17)   z.used := {};
18) endif
```

1 行目と 2 行目は b が何にもマッチせず、 b の前の記号が y にマッチした場合を扱う。3 行目から 7 行目までは b が B の中の以前にマッチしなかった記号にマッチし、 y の中にある他の記号にマッチした場合を扱う。8 行目から 12 行目は、さらに良いアソシエーションが b と B とのマッチングによって得られ、他の記号が w にあるとしてマッチするかどうかテストする。最後に 14 行目から 18 行目は b が何にもマッチせず、 x にある他の記号にマッチする時を考える。

伴奏

マッチが起きた時 Accompanist は 2 つのルーチンを呼び出す。

最初のルーチンは change_virt_time と呼ばれ、仮想時計の時刻合わせを行ない、現在鳴っている音符を変更する。2 番目のものは adjust_clock_speed と呼ばれ、ソロ演奏のテンポを反映するために伴奏演奏のテンポを変化させる。これらのタスクは完全に独立に扱われ、それぞれ仮想時計の位置と速度を調節する。

仮想時間の変更 (change_virt_time)

もしマッチが「連続している (in sequence) 」のなら、伴奏は単に独奏者よりも遅く、あるいは早く演奏する。そのとき音符をスキップしたり、同じ音符を二度以上演奏するという危険を犯して伴奏楽譜中でジャンプするよりは、単に順番に伴奏イベントを演奏し続けた方が良い。もし遅れているならば追い付くためにスキップすべきイベントを素早く演奏し、もし追い越しているのなら、演奏している音符を（もしあるなら）音符が消されるべき仮想時間に達するまで演奏し続ける。しかし独奏者が大きな時間変化を行なった時、その時間がある閾値（我々は 2 秒を用いている）よりも大きいならば、追い付くために急いだり、止まって待ったりするよりは伴奏を正しい場所にスキップするほうが良い。

もしマッチが連続でないなら Accompanist が信じている独奏者の楽譜中の位置はおそらく間違っているため、独奏者に即座に同期する。当然のことながら、仮想時計の調節と伴奏楽譜中で演奏されるべき次のイベントへのポインタは時間変化を反映するように更新される。

時計の速度調節 (adjust_clock_speed)

Matcher はマッチを報告した時、マッチしたソロ・イベントの時刻を Accompanist に与え、この時刻が現在の仮想時刻になる。ACS は実時間時計を参照することにより、仮想時間 vs. 実時間グラフ上に点を得る。ACS は最近のいくつかの「マッチ・ポイント」のテーブルを保持する。(我々はテーブルに 4 ポイントを保持する。) ACS が呼ばれる時刻ごとに新しいポイントがテーブルに付け加えられる。もしテーブルがすでに一杯であるなら、新しいポイントを入れるべき場所を作るために最も古いポイントがテーブルから削除される。ポイントが加えられた後、テーブルが一杯であるなら、 ACS は s をテーブル中の点の範囲におけるグラフの傾きでリセットする。よりソフィスティケートされたアプローチとして、これらの点に直線を近似し、s をその傾きにセットする。

もし順序が狂ってマッチが検出されたら、この点はそれよりも前のものとは関係がなくなるので、 ACS はその点をテーブル中に蓄積し、この点を除いてテーブルを空にする。

仮想時間と時計の速度調節アルゴリズムを頻繁に呼ぶことができれば、伴奏をソロ演奏と同期することができる。しかし、もし Matcher が独奏者の追跡に失敗したり、独奏者が演奏をやめてしまったら、 Matcher は Accompanist にマッチを報告するのをやめてしまう。このとき仮想時計は同じように時を刻み続け、独奏者が演奏を合わせていないという事実に気づかずに伴奏は最後に計算されたテンポで演奏を続けてしまう。この振舞いを防ぐため Accompanist は逃亡の検出 (runaway detection) を備える。そのため、 Matcher はマッチが報告される時刻ごとに Accompanist に予測される次のイベントの仮想時刻を教える。もし次のイベントに対する予測時間よりも十分長く仮想時間 (我々のプログラムでは 2 秒) が経過してもマッチが報告されなかったら逃亡が宣言される。そのとき現在鳴っているすべての音符を消し、仮想時計の進行を止める。逃亡はマッチが検出されるまで続く。マッチが検出された時、仮想時計はリセットされ、伴奏は継続される。伴奏は Matcher が新しい位置を報告するとすぐに再開される。

コンピュータ伴奏の新しい技術

R. B. Dannenberg らは 1988 年の ICMC において、従来の伴奏システム [Dan84, BD85] をいくつかの点で改良する論文を発表した [DM88] 。その一つは演奏が二通りに解釈される可能性がある時 matcher をもう一つ使用することである。もう一つの改良点は matcher からのデータが疑わしい時には訂正する働きを遅らせることである。また装飾音、トリル、グリッサンドを取り扱うことも可能にした。さらにオクターブの違いを無視し、独奏者の入力情報を 2 バイトで表現することにより、マッチング処理の効率化をはかっている。

複数のマッチャー

matcher は常に窓 (window) と呼ばれる楽譜の一部分だけを考える。もしソロ演奏家が窓の中で演奏しなかったら matcher はソロ演奏家を追跡することができなくなってしまう。解決法は複数の matcher を戦略的にマッチしそうな場所におくことである。

matcher は時間のコストが高いモジュールなので常に多くの matcher が動いているのは望ましくない。独奏者の位置がある程度不確かな時に限りもう一つの matcher を呼び出し、その窓を代わりの位置に置く。独奏者の位置に関して一つ以上の妥当な推定がある時に新しい matcher を作成する。これはソロ演奏家が演奏をやめた時や多くの余分な音を演奏した時に起こる。このシステムでは伴奏の「仮想時間」と matcher が見ている窓の「仮想時間」の範囲とをチェックしている。もし現在の「仮想時間」がその範囲になければ新しい matcher をその「仮想時間」が現在の伴奏の仮想時間に等しくなるような位置に作成する。

一つの matcher がマッチを発見した時に他の matcher は終了する。

トリルとグリッサンド

matcher の仕事は基本的に「演奏」と「楽譜」との一対一マッピングを見つけることである。しかしこれはトリルやグリッサンドの場合には正しくはない。なぜならこれらの装飾音は演奏家に依存していて、楽譜では正確に記述されていないからである。この種の予測不能性を処理するためには「多対一マッピング」を見つける機構を準備する必要がある。

プリプロセッサ[8]は楽譜情報を利用することにより、トリルを特別の複合イベントに変換する。プリプロセッサはトリルを単一のイベントに変換するので matcher はデータが特別なのかどうかということを考える必要はなくなる。プリプロセッサは有限状態機械からなり、「トリル」や「グリッサンド」などの特別の記号を楽譜中に検出すると内部状態を「ノーマル (normal) 」から「トリル／グリッサンド (trill/glissando) 」へと変化させる。予測されるトリル／グリッサンドの終了時刻のような情報も楽譜から同時に得る。このようにプリプロセッサは以下の制約のどれか一つが満足されるまでトリル／グリッサンド・モードの中にいる。その制約は、プリプロセッサが長い音符を得た時、その装飾音の終了時刻になった時、トリル／グリッサンドの次の音符がその予測される時刻の近くで演奏された時の 3 つである。

プリプロセッサはトリル／グリッサンド状態に入った時、特定のシンボルを matcher に送る。プリプロセッサはその状態にある限り matcher に対しては情報を送らない。matcher はマッチを得るごとに次の音を予測できる。もしこれがトリルやグリッサンドを示している特別のシンボルなら matcher はプリプロセッサにそれを教える。これはプリプロセッサの中の状態遷移を引き起こす。

遅延決定

新たに演奏された音符が以前のすべてのマッチよりも長いマッチになる時には常に matcher はその位置を accompanist に報告するが、しばしば プリプロセッサはトリルやグリッサンドなどの入力を正しく取り扱うことができず、いくらかの余分な音を matcher に送ってしまう。このような場合間違ったマッチを見つける可能性が若干ある。このような思いがけないマッチングを避けるために matcher からの報告をすべて信じてはいけない。Accomapnist は、もし最後に報告されたのが $n-1$ 番目のマッチで、その後 n 番目の音符がマッチしたならそれを信用することができる。このようにマッチが連続的な場合は accompanist が演奏すべき場所は信頼を増すことができる。しかし、もし新しいマッチが最後のマッチに連続していなかったなら、最近演奏家が何らかの間違いをしたということが予想されるので、システムはそのマッチに疑いを持つ。そのような状況の時には matcher は accompanist への報告を一定の遅延時間内で遅らせる。プリプロセッサは遅延時間内に音符を受けとったら、何が間違いであったのかを認識し、遅延報告をキャンセルする。もしキャンセルするようなことが起こらなければ報告をそのまま accompanist へ送る。accompanist はこの遅延を考慮し、それを補う。

オクターブの等価性

オクターブの違いは無視する。このようにすることによってソリストはどんな音域でも演奏することが可能となった。また closed chord として記述されていたとしても opened chord として演奏することができる。しかしオクターブ情報を落とすことにより、例えば C3 と C4 を同じ音として取り扱ってしまい、間違ったマッチを得る危険が若干ある。最も良いアプローチは音楽と演奏に基づいて考えることであるが、この選択によって以下で記述するような良い結果を得ることができた。

[8]このモジュールは MIDI-IN からの入力を処理し、時間情報を用いて複合イベントを作成する。もし短時間に二つあるいはそれ以上のイベントの入力があった場合 プリプロセッサはそれらを一緒にして複合イベントを作成する。またトリルやグリッサンドを検出・処理する。

ビット・ベクトルの使用

ポリフォニック音符を処理するために集合を表すビットベクトルを用いる。ベクトルは 2 バイト整数であり、その一番左のビットは「 C 」音に相当し、その隣のビットは「 C♯/D♭ 」音に相当する。このデータ構造を用いることにより 2 バイトですべての音符の組合せを表現することができる。マッチング計算は時間がかかるので音符データとその操作をできるだけ簡単かつ高速にすることが非常に重要となる。図 4.25 を見て欲しい。楽譜イベントと演奏イベントとのビットごとの「排他的 or 」をとることにより「 result1 」を得る。これは両者の違いをビットで表現している。それから result1 と楽譜との「 and 」をとる。これを「 result2 」とすると、これは楽譜のどの音符が演奏されなかったかを示す。 result1 から演奏されなかった音符と余分に演奏

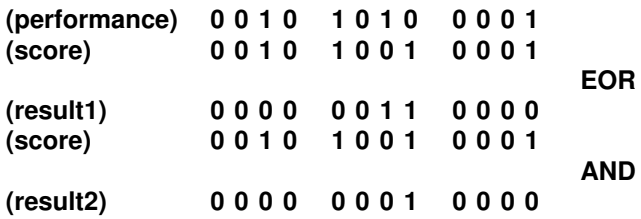


図 4.25: ビット操作

された音符とが分かる。余分な音は装飾音符かもしれないので、余分な音符は無視する。これは「 and 」操作により実現される。そうすることにより演奏されなかった音符だけを得ることができる。（これは若干簡単化でき、 $result2 = score \wedge \overline{performance}$ となる。）ビットを数えあげることにより、演奏されなかった音符の数を得ることができる。 もしこの数が楽譜イベントの音

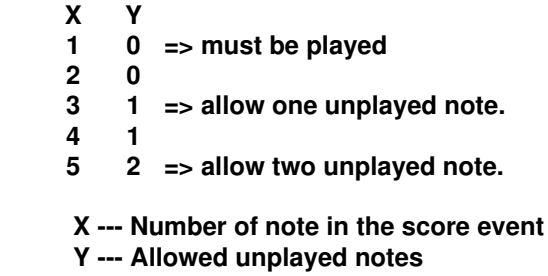


図 4.26: 演奏されなかった音符を許可するテーブル

符数から決定される定数よりも小さかったらそれをマッチとして扱う（図 4.26 参照） 。このようにして伴奏システムは、演奏家が難しい和音を不完全に演奏しても許容することができる。

これらの操作の長所は matcher での集合演算を効率良く行なえるということである。またこれらの操作は装飾音符を効率的に除去し、演奏者にペナルティなく余分な音を演奏することを許す。

リアルタイム即興演奏追従

上記の伴奏システム [Dan84, BD85, DM88] は独奏者の楽譜情報をあらかじめ与えているが、R. B. Dannenberg らは 1987 年の ICMC において独奏者の楽譜を与えずに、即興演奏を追跡する方法を発表した [DMR87] 。この研究ではジャズのソロの拍の位置を推測し、追従できるコンピュータ・システムを構築するための技術を調査している。

拍追従

ブルースのソロに追従するため、たとえば演奏とほぼ同期した八分音符（我々は八分音符を追跡することにする）をカウントできる「拍追従器 (beat follower) 」が必要である。

簡単なリアルタイム・フット・タッパー

拍追跡器 (beat tracker) の状態は 5 つの数字 (T,B,S,W,D) によって表現される。実数 T と整数 B はもっとも最近作られた位置の秒単位の時間と拍単位の位置を表す。実数 S と W は現在の拍の値 D=S/W を生じる。我々が現在の拍の値に対して単一の数よりもむしろ同次座標系を用いる理由は二つの数 S と W を連続した統計のままにしておくことができることである。S は拍間隔の現在の合計で W は現在の重みである。

状態 (T,B,S,W,D) にある拍追跡器、予測器は本質的に拍時間 (metrical times) B+1, B+2, ... と物理的時間 (physical times) T+D, T+2*D, ... , でのイベントを予測する方法である。

我々はいくつかの用語を必要とする。もし (D1 * a + b) < D2 であるなら D1 は D2 よりも非常に短いと言い、D1 ≪ D2 と記述することにする。ここで a は 1.1 のような 1 よりも大きい定数であり、 b は .1 秒というような小さい時間間隔である。もし D1 ≪ D2 でも D1 ≫ D2 でもないのなら D1 は D2 に大体等しいと言う (D1 == D2) 。もし、ある絶対的な閾値（例えば .05 秒）よりも短いか、あるいはもし先行する音符よりも短いのなら、その音符は「弱い (weak) 」と言う。もしある音符が弱くないのなら、その音符は「健全である (healthy) 」と言う。もしある音符が強い最初の音符であるか、あるいは先行する音符がそれ自身の音符よりも十分に短いのなら、その音符は「アクセントのある (accected) 」と言う。ここで二つの主なアルゴリズムを述べる。その一つは予測器 (predictor) を作成することであり、もう一つはその状態を更新することである。

予測器を作成するのに用いられる方法は「リズム・アラインメント (rhythmic alignment) 」を見分けることである。すなわち、その立ち上がり時刻が T1, T2, T3 である 3 つの健全な音符 N1, N2, N3 の連続は、 (T2 − T1) == (T3 − T2) となるような算術数列である。N1 の N2 に対する関係と N2 の N3 に対する関係は、 N(i + 1) は N(i) の直後に続く音符であるか、 N(i) に続く最初のアクセントのある音符のどちらかであるということである。

状態は (T1, B1=0, S=0.0, W=0.0, D) で初期化される。拍時間はそのようなアラインメント（拍の最初の感覚）が見つかるまでは数え始めないので、B1 は任意であるが、我々はこれを 0 とする。S/W=0.0/0.0 は一般には不定であるため、 D の値の選択は明らかではない。初期拍の選択は、二つの大体等しい時間間隔 (T2 − T1) と (T3 − T2) の平均 A を取り、D=A/round(A/E) となるような予測される八分音符 E の範囲にそれを近付けることによって行なわれる。例えばもし、観測された最初の 2 つの音符が .4 と .46 の時間間隔を持ち、その平均が .43 、 E が .20 なら、我々は D = .43/2 から始め、最初の二つの音符を四分音符として扱うということになる。

「更新」は新しい音符を認識することに等しく、それに応じてテンポを調節する。我々がアラインメント N1, N2, N3 を認識した後、 N2, N3 の順で認識するため、 N1 で新しい予測器をセット・アップする。その後は、予測器はすべての「健全な」音符によって更新される。 (T,B,W,S,D) を現在の状態、 T' が新しい健全な音符の立ち上がり時刻とし、

$$\Delta T = T' - T$$
$$\Delta B = \Delta T \ / \ D$$
$$DELTA = \mathrm{abs}(\Delta B - \mathrm{round}(\Delta B))$$
$$CONFIDENCE = 1 - 2 * DELTA$$
$$\Delta B = \mathrm{round}(\Delta B)$$
$$DECAY = .9 ** (-\Delta B)$$

とすると、状態は以下のように更新される。

T = T + ΔT i.e. T'
B = B + ΔB
W = W * DECAY + ΔB * CONFIDENCE
S = S * DECAY + ΔT * CONFIDENCE

リアルタイム・ビート・トラッキング

上で述べたトラッキング方法 [DMR87] では現在の拍時刻と位置の解釈を一つしか保持しなかった。1990 年の ICMC で P. E. Allen らは、複数の解釈を保持して、ビーム・サーチを用いる新しいモデルを提案している [AD90] 。拍のタイミングと位置の予測は、考慮されている複数の解釈のうちで最も信頼できるものにより行なっている。

ビート・トラッキングの初期モデル

Mont-Reynaud が開発したリアルタイム方法 [DMR87] におけるヒストリと信頼機構をパラメータ化した。ヒストリ機構は過去のテンポの重み付けに指数ディケイを用いている。ディケイ値は拍追跡器の敏感さと安定性に直接的な影響を及ぼす。ディケイ 100% では追跡器はすべての履歴を無視し、最新の音符データのみにしたがって応答するであろう。そのとき急速なテンポ変動を許し、追跡器は不安定になってしまう。ディケイ 0% では、追跡器は履歴のそれぞれのものを現在の状態と等しく考慮し、そのため最新の音符データの考慮はわずかで応答は遅くなるであろう。ディケイの値は調査すべき一つのパラメータである。

二番目のパラメータ化は信頼の計算に適用する。信頼は与えられた音符の立ち上がりが強拍上に生じたと拍追跡器が信じる確からしさ、間接的には立ち上がり時間に応じてテンポが変化すべき量を表すのに役立つ。以前の信頼関数はのこぎり歯関数であり、予測された強拍で最大値 1.0 、隣合わせの予測された二つの強拍のちょうど真中で最小値 0.0 を取り、その傾きは 1.0 か -1.0 のどちらかである。我々はオリジナルの信頼の計算を採用し、それを指数パラメータにする。（このようにすると 1.0 のパラメータ値で、最初の方法と同じ信頼計算となる。）このことにより、予測された強拍からの立ち上がりの距離の関数として音符の立ち上がりが新しいテンポに与える影響を調整することができるようになる。ここで我々は不正確な強拍の音符の許容と正確に演奏された強拍でない音符の廃棄との間の歩み寄りを捜している。

新しいビート・トラッキング方法

我々は周期と位相の二つの面を含むように「拍」の音楽的概念を拡張する。音楽演奏における拍の周期は二つの連続する拍の時間間隔（テンポの逆数）であり、拍の位相は演奏時間に関して拍が発生する時刻を決定する。拍追跡器は少なくとも現在認知された拍の位相と周期を含む状態情報を持たなければならない。

従来の方法の困難の一つは、拍追跡器は現在の拍周期と位相の信念を示す唯一の新しい状態を決定する必要があるということである。もしある点で間違いを犯してしまったら、回復することは決してできない。

拍追跡器が同時にいくつかの可能な状態を考慮できるようにするため、我々はリアルタイム・ビーム・サーチを用いる方法を構築した。その方法は以前の方法と同様にヒストリ機構を用いるが、それをガイドするのにもはや信頼の大きさだけに頼ったりしない。そのかわりに確実性の大きさが用いられる。そのためこれまでに直面した演奏の最も信頼できる解釈を表現する一組のアクティヴな状態が常に存在する。それぞれの状態は、その状態が表現する特定の解釈の確実性を測定するヒューリスティック評価関数によって決定される確実性の値を持つ。最も確実な解釈とみなされる状態は、最も確実な新しい状態が見つかるまで、拍の位相と周期の予測を行なう基礎として用いられる。演奏中に新しい音符の立ち上がりに遭遇した時、現時点での複数のアクティ

ヴな状態はそれぞれ、その立ち上がりから生じる複数の可能な解釈とアクティヴな状態の解釈とを表すいくつかの新しい状態に拡張される（図 4.27 参照）。音楽的な意味のない解釈はすぐに捨てられる。残っている状態の集合は次の拡張のために新しいアクティヴ状態になる。

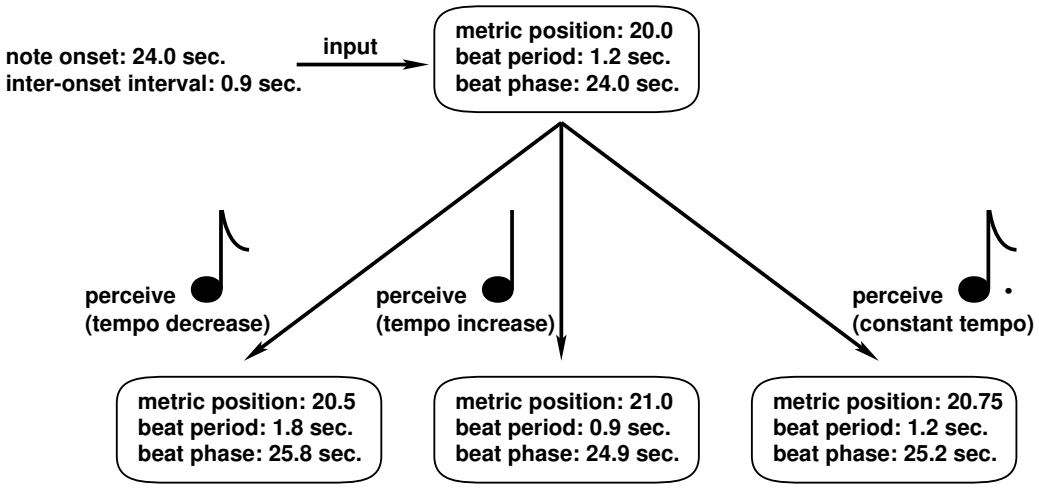


図 4.27: 状態拡張の簡単な例

ここで提案されている手法はリアルタイムの方法で、たくさんの解釈が永久に続くので、我々は検索の大きさを減少するため刈り込み技術を適用しなければならない。最初の刈り込み方法はすでに述べた音楽的に意味のない解釈の状態を破棄することである。二番目の技術は状態の確実性と拡張を行なう順序を意味する。我々は新しい入力が準備されるまで拡張を行ないたいので、最も確実なものを最初に拡張し、確実性に従う順番で状態を拡張する。このようにして拡張している間に、もし新しい入力が処理される準備が出来ていることが分かり、最も信頼できるものがすでに拡張されたということを知っているなら、アクティヴ状態の拡張を即座に止めることが出来る。（もちろん、最も信頼できる少なくともいくつかの状態を拡張するため、入力の間には十分な時間があるものと仮定する。）三番目の刈り込み方法は、類似した解釈を表現する複数の状態を一つの解釈を表現するのに一つの状態に合併することである。四番目の検索刈り込みの技術は、一つのアクティブな状態の拡張から生じる新しい解釈の数を制限することによって枝分かれの要因を制限することである。その拡張の数は任意の静的カット・オフによって制限する。一つの状態からの拡張は、結果として生じる解釈が最も小さいテンポ変化になるような順番で実行される。

4.3.3 マサチューセッツ工科大学メディア・ラボでの研究

この節では M.I.T. の B. Vercoe を中心とする伴奏システム研究について述べる。Vercoe も Dannenberg と同様に 1984 年の ICMC で彼の最初の伴奏システムを発表している [Ver84] 。Vercoe は Dannenberg とは異なり、独奏者の入力情報としてアコースティック楽器に対する伴奏システムを提案している。また Vercoe はリハーサルの考えを最初の論文 [Ver84] から主張しており、実際に 1985 年にそのアイデアを発表している [VP85] 。以下、それらの研究について述べる。

シンセティック・パフォーマー

B. Vercoe は 1984 年の ICMC でシンセティック・パフォーマー（人工演奏家）と呼ばれる伴奏システムに関する研究を発表している [Ver84] 。彼はその論文の序文において研究の目標を以下のように明確に述べている。

> 演奏グループの誰かを（コンピュータモデルとしての）シンセティック・パフォーマー (synthetic performer) によって置き換えても、残りのライブ演奏家がその違いに気が付かない位十分にライブ・アンサンブル演奏の dynamics を理解すること

シンセティック・パフォーマー

シンセティック・パフォーマーをモデル化する制御構造は以下の 3 つの主要部分からなる。

I 聴く (LISTEN) :
1. 入力されるライブ・イベントを認識し、解析する。
2. テンポ、楽譜位置、音量などを抽出する。

II 演奏する (PERFORM) :
1. 新しい演奏テンポ、音量、フレージングをセットする。
2. コンピュータ演奏を行なう。

III 学習する (LEARN) :
1. 予測されなかった Listen の振舞い、 Perform の難点を観察する。
2. 将来の準備のためフル・スコアに付け加える。

シンセティック・リスナー

プロのフルーティストによって演奏されたイベントのシーケンスを認識し、解析するための戦略として、指使いの情報と楽譜という二つの付加的なデータを用いる。フルーティスト Larry Beauregard のキーの上に取り付けられた光学センサーの並びによって可能なピッチのリストは 3 つに減らされる。そして音響信号を適当なフィルターに通すことによって、 4X[9] は大体 35 ミリ秒でその 3 つの曖昧性を解消することができる。

ライブ演奏家による可能な逸脱を許すように、ピッチと時刻の情報はともに楽譜の要素に写像される。音楽認識 (music recognition) は、若干のリズムとピッチのフォールト・トレランスと合わせて、かなりのリズム伸縮性を必要とすることが分かった。ふたつ、あるいはそれ以上の音符のエラーは自動的に、より厳密なパターンマッチングを、あるいはしばしば大規模な再配置 (extensive relocation) を引き起こすべきである。完全に認識できないイベントに対する最も良い戦略は、何か認識可能なことが生じるまで現在の方針を維持することである。

このようにマッチしたシーケンスからテンポ感を抽出することは、さらにアゴーギグの時間シフトのような効果の吸収を必要とする。一拍の中で観測される時間シフトは位置によって重み付けされる。拍の初めの方の部分の変更は一般的に後ろの方で生じるものよりもテンポの重要性を持たない。 Listen で一度新しいテンポが決定され、適当であることが分かると、それはシンセティック・パフォーマーに知らされる。

演奏の性質

シンセティック・パフォーマーの正しいテンポ決定には 2 つのレベルの動作が必要である。その一つは 12 ミリ秒間隔で（拍の長さの形式で）報告された Listen のテンポをサンプリングし、beat-bins に蓄えることである。これによりテンポはかなり高い精度で時間上で積分される。そ

[9] IRCAM のリアルタイム・オーディオ・プロセッサ

の後、大体 200 ミリ秒に一度、 beat-bins はライブ演奏家の明確な楽譜位置を決定するのに用いられる。これはシンセティック・パフォーマーの楽譜位置と比較され、自然に追い付く動作を決定する。1 秒間にちょうど 5 回程度の決定は演奏者がこの種のことを行なう方法を十分に表しているようである。

演奏の生理学的機能をモデル化することは若干トリッキーである。私の見解では、意図される演奏イベントは、取り消すことが不可能な生理学的制御対象に突然変換される行動のほんの一瞬前まで、厳密な拍子の表現 (strict metrical terms) のままである。演奏の生理学的ジェスチャーに人間が必要とする時間は人間とそのデバイスに依存するが大体 1/10 秒間であろう。演奏モデルでは、楽譜イベントがこの閾値を一度またぐと事実上、行動対象モジュールの一群に爆発する。それらのモジュールは、それぞれがイベントのいくつかの局面、すなわち、立ち上がり時間 (rise time) 、瞬間的効果 (transient effects) 、周波数・音量カーブ (frequency and loudness curves) を表し、それぞれ CPU サービスを必要とする。プロセッサは優先キューにこれらの要求を与え、シンセサイザー自身の中に概念的に備わっている時刻付きバッファへ制御データを送る。

学習による改善

上で述べた制御プロセスのネットワークは学習された反応ではなく、厳密にいえば「本能的な」ものである神経有機体 (neural organism) をモデル化することとみなすことができる。シンセティック・パフォーマーの最も苛酷なテストは、あたかも演奏会ステージで初見で演奏するように、過去に集められた情報を持たない状況でどの程度巧みに演奏できるかどうかである。

最初、 シンセティック・パフォーマーの学習戦略を含めて考えていたが、そのプログラム・コードは通常は利用していない。何故なら、それは本能的モデルの開発とテストを複雑にするからである。これらの局面が、適度に高いレベルでの robustness をどうにかして示せば、適応的な学習応答機構をさらに開発する予定である。特定の楽譜と認識されたライブ入力に学習応答を付け加えることはシンセティック・パフォーマーの重要な音楽的才能を大いに改良することが期待される。

シンセティック・リハーサル

上で述べたシンセティック・パフォーマー [Ver84] はかなり敏感なライブ・トラッキングのみに頼っており、過去の経験の利益を用いる学習能力や訓練能力がなかった。そのためシンセティック・パフォーマーは基本的にコンサート・ステージでは「常に初見視奏」であった。B. Vercoe らは 1985 年の ICMC で、リハーサルを行なうことによってシンセティック・パフォーマーを訓練する方法を発表した [VP85] 。

経験の記憶

リハーサルを続いて起こる予想の決定要因としてみなすための最初のステップは、リハーサル・データが有用になるまでたくさん保存することである。時間領域では、その情報にはリズムの逸脱とテンポのゆがみという二つの主要な種類がある。それら二つは別個のものではなく、局所的文脈では、お互いに別のものに間違われ得る。認知的トリックは有効なテンポと現在の楽譜位置を常に判断し、残りのゆがみをリズムのものとみなす。入力される各音符が楽譜中で認識されるにつれて、予測されるよりもイベントが早いか遅いかを表す拍の分数をその演奏記録の中に書き込む。この量は現在確信している楽譜位置に関して厳密に異なるインデックスである。

演奏前のメモリ・マッサージ

リハーサルの効果は、以後の演奏の楽譜の輪郭を描くことを助ける新しい意味概念の構成を許

すことである。我々はテンポ検出を改善するためにリハーサル・データを用いる。例えば、楽譜のそれぞれの音符の演奏が抽出されたテンポにどのように関係する傾向があるかというような統計を集めることによって。それぞれの演奏の後、この演奏家の動作時刻のイベント・リストを得る。リハーサルされず、リアルタイムでない完全なイベント・リストのテンポ解析を行ない、実時間の関数 ϕ として楽譜位置を与えることによって楽譜中にこの経験を組み込む。我々はこれをリアルタイムに計算しないため、ϕ のより正確な値を得るために未来を調べることが可能である。それぞれの楽譜イベント (s_i, x_i) に対して標本平均 m_i 、観測された音符と予測されたものとの間の楽譜位置の食い違いの標準偏差 σ_i 、あるいは、 $s_i - \phi(t_j)$ を集める。ここで (i, j) は実際の楽譜と観測されたものとの間の対応である。

4.3.4 コネティカット大学での研究

コネティカット大学の Baird らは人間の演奏家はメロディの「モチーフ」を追跡しているという考えに基づき、音符をユニットとし、それらをまとめたパターン（音楽的モチーフ）ごとのマッチングを再帰的に取る方法を提案した [BBZ89] 。またその後、それを Macintosh 上のトランスピュータを用いて、複数の人間の演奏を処理し、それに合わせて演奏を行なうシステムを紹介している [Bai91] 。

Artificially Intelligent Computer Performer

B. Baird らは、1989 年の ICMC で彼の伴奏システム AICP (Artificially Intelligent Computer Performer) を発表している [BBZ89] 。AICP はコンピュータが演奏家の演奏を聴き、それに伴奏をつけることを可能にするソフトウェア・プログラムである。この論文では彼らの提案するパターン・マッチング・アルゴリズムが中心に述べられている。

アルゴリズム

演奏家は聴取者と同様、メロディの「モチーフ」を追跡していると考えられる。「音楽的モチーフ」は AICP のトラッキング・アルゴリズムの基礎をなす。楽譜とライブ演奏とを比較するため、ユニットは一つ一つのピッチと休符、及びその長さからなる。ユニットの連続はパターンを構成し、演奏家が演奏するにつれてそのパターンが作られてゆく。トラッキング・アルゴリズムはこのパターンと楽譜のパターンとのベスト・マッチを見つける。それは真中が現在の演奏家の位置になるような楽譜中の窓を調べる。パターンのユニットに対して、3 つの異なった種類のマッチが許される。すなわち、 verbatim match （そのままのマッチ）、 amalgamated match （融合されるマッチ）、 held through match （継続するマッチ）である。最後の二つはともに演奏エラーに相当する。再帰関数によって、これらの種類のマッチとそれぞれのパターンのペアとの音楽的に可能なすべての組合せがチェックされる。

verbatim match ライブ演奏家によって演奏された各ユニットと楽譜の各ユニットとの 1 対 1 対応。コストはピッチの近さと長さの違いに基づいて割り当てられる（図 4.28 example 1a 参照）。

amalgamated match ライブ演奏家の 2 つの入力されたピッチを見て、それらを 1 つのユニットにマージする。これはライブ演奏家がピッチのミスをおかしたと考えられ、2 つのアタックは 1 つのユニットを意図したものであったとする。結果として融合されたユニットは、コストに若干のペナルティを加えて verbatim match と同様に扱われる（図 4.28 example 1b 参照）。

held through match 楽譜の 2 つの音符を演奏の 1 つの音符にマッチすると考える。これは演奏家が新しい音符を演奏するのに失敗したが、その代わりに前の音符を次の音符まで伸

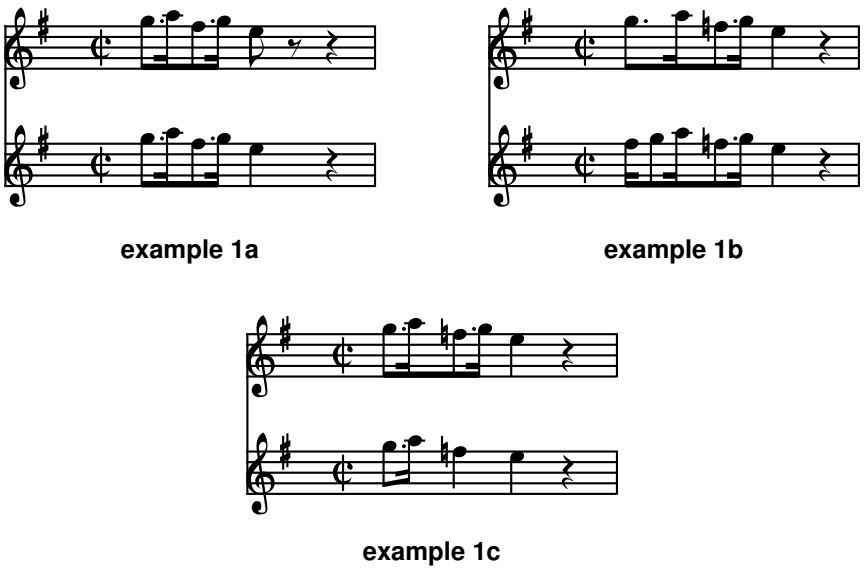


図 4.28: マッチングの例

ばしているということである。楽譜の 2 つのユニットは 1 つのものとして扱われ、そのピッチは最初の音符と同じ、長さは 2 つの和とする。このユニットはコストに若干のペナルティを加えて verbatim match と同様に扱われる（図 4.28 example 1c 参照）。

AICP と並列処理

B. Baird は 2 年後 1991 年の ICMC で、 AICP の並列処理に関して発表している [Bai91] 。そこでは Macintosh 上のトランスピュータによる並列処理で、コンピュータ・パフォーマーは何人かのライブ演奏家の演奏を同時に聴き、その入力と楽譜とをマッチし、それぞれの位置を推定して応答する。

AICP 上での並列処理

AICP は 8 チャンネルまでの情報を受け取り、 8 つまでのトランスピュータを扱える。楽譜は標準 MIDI 形式でプログラムの中に読み込まれ、チャンネルは楽譜を特定することにより活性化される。活性化された各チャンネルはメニュー選択によって、ライブ演奏家やファイルからの入力、コンピュータ演奏、オフのどれかが可能になる。ライブ演奏中、シンセサイザーを通してライブ演奏家から、あるいはファイルからの入力があるかもしれない。ファイルからの入力は実際にはコンピュータによって演奏されるが、人間による演奏と同じように扱われる。ライブ演奏の間、コンピュータは残りのアクティブなチャンネルすべてを演奏する。演奏のリプレイのために、あるいは将来の演奏の入力として用いるために、すべてのライブ演奏はプログラムによって録音され記憶されている。ファイルからの「ライブ」演奏は標準 MIDI 形式、あるいはユーザによって保存されたファイルから読み込まれる。プログラムはライブ演奏、演奏リプレイ、楽譜の選択されたパートだけの演奏が可能である。

それぞれのトランスピュータは一人のライブ演奏家チャンネルに相当する。このチャンネルに対する楽譜は演奏が始まる前にトランスピュータに送られる。演奏が進むに連れて、演奏家からの音楽入力、テンポ変化、その他の情報が適当なトランスピュータに送られる。それぞれのトランスピュータは、そのチャンネルを用いている演奏家の適当な楽譜位置を決定するためにパターン・マッチを行ない、その情報を Macintosh に通信する。

Macintosh はトランスピュータからの情報を受け取り、適当なコンピュータの応答を決定するためにすべての情報を調和させる。音楽家はすべての演奏を聴くけれども、一人の演奏家に追従する傾向がある。その演奏家の選択は演奏家の楽器の重要さ（例えば弦楽四重奏ならば第一ヴァイオリン奏者）、演奏家の音楽ライン（メロディやビート・キープ）や演奏が正しいという信頼によって決定される。AICP はそれぞれのトランスピュータから演奏家の楽譜上の位置の推定ばかりでなく、その推定の確実性を示すコストを受け取る。このコスト、演奏があるかないか、楽器の重要さはすべて、コンピュータがその応答を決定するのを助けるのに用いられる。

4.3.5 クィーンズ大学での研究

クィーンズ大学の Driesse はリズム認知に関するさまざま規則を適用することによって、楽譜を与えられていない演奏のテンポ・トラッキング・システムについて発表している [Dri91] 。

リズムの性質の解析規則を用いたリアルタイム・テンポ・トラッキング

検出規則

検出器 (detector) は、いくらかの音楽的知識を基礎として、どの音符が拍上で起こっているのかを決定しなければならない。検出器は音符のピッチ（キー番号：key number ）、音符の立ち上がり時刻 (note onset times) 、音符の継続時間 (note duration) 、アタック・ダイナミクス（キー・ヴェロシティ：volocity ）を認識するか、それにアクセスすることができる。音楽的知識の規則はリズム特性の強度を「なし (absent) 」から「強い (strong) 」までの値としてスコア付けする。その値は音符が拍上に乗っている可能性の高さとして解釈することができる。

インプリメントのため、以下の 5 つの規則を選択する。

1. 相対的に長い音符は拍の上で始まりやすい。
2. 相対的に音量の大きい音符は拍の上で始まりやすい。
3. 低い音は高い音よりも拍の上で始まりやすい。
4. ほぼ同時に起こるいくつかの音符は拍を示しやすい。
5. 現在鳴っている音符の間の和声的関係の変化は拍を示しやすい。

音符が拍の上にある可能性を計算するため、これらの規則に相対的な優先度を付けて適用する。

予測規則

テンポ変化を数学的にモデル化しようとするより、可能なテンポ変化の限界を課すたくさんの規則を形式化する。これらの規則はまた、演奏家による潜在的なタイミング・エラーを説明できなければならない。

予測器は最近の数小節で演奏されたものだけしか解析しないが、現在の拍までの予測された拍、検出された拍のすべての時間情報を用いることができる。最初に予測器はランダムなタイミング・エラーの効果を減らすために最近の 1 〜 2 小節の連続している拍の検出された時間間隔（拍周期：beat periods ）の重み付け平均を計算する。次に初期予測を行なうため、この平均拍周期を最後に検出された拍の時刻に加える。平均によってテンポ変化が緩やかになるので、予測器はさらに拍周期の間の差の平均を加える。

いくつかの付加的な規則が予測器に対して適用される。

・もし予測器によって提案された拍が急激な変化を引き起こすのなら予測器はそれを破棄する。

・もし予測器が拍を置くのに失敗したら以前の予測が正しいと仮定する。

- もし突然の変化を予測するのなら、変化の量を制限する。
- もしコンピュータが一様にリードしたり遅れたりするのであれば、それを償う。

4.3.6 早稲田大学（大照研究室）での研究

早稲田大学の直井らは、あらかじめ楽譜情報を与える代わりに、実際に伴奏パートの演奏を人間の演奏により入力し、その情報を用いて伴奏システムを作成している [直井 89, 直井 90]。井川らは、独奏者と伴奏システムとの間の相互作用をモデル化している [井川 90] 。また井上らは、人間の歌声を入力とした伴奏システムを提案し [井上 92b] 、その後、任意の調で曲の途中から歌い出しても、その場所から独奏者と同じ調で演奏を行なう拡張も行なっている [井上 92a] 。

実時間拍検出機能を用いた自動伴奏システム

楽譜をもとにした演奏では、タイミング、速度など楽譜に現われない微妙な情報を付加することは難しい。直井らは伴奏情報を直接演奏により入力し、その時間的なニュアンスも生かせる一つの自動伴奏システムを開発した [直井 89, 直井 90] 。これは、あらかじめ伴奏パートを演奏により入力しておき、そのあと、この旋律パートを演奏すると、旋律パートに応じて伴奏パートが自動追従するものである。

このシステムでは、まず最初に伴奏パートを単独で入力する。その伴奏パートのデータを、すでに報告されている自動拍検出機能 [大照 88] を用いて解析し、演奏時刻と拍位置（音長の最小単位を基準とした位置）との対応を調べる（演奏を音長の最小単位でクオンタイズするようなものと考えれば良い）。得られたこの情報は、従来の研究における伴奏パートの楽譜情報と考えることが出来る。次に独奏パートを演奏し、それと同時に伴奏パートも演奏する。そのとき独奏パートの演奏から拍を検出して、伴奏パートの速度、位相を制御して追従する。

相互作用モデルによる実時間適応自動伴奏

自動伴奏の研究は従来専ら機械が人間に追従する形で進められてきたが、本来自動伴奏は人間と機械の同期演奏を試みるもので両者による演奏は互いに影響を及ぼしあい構成される。そこで井川らは、独奏者と自動伴奏との間に相互作用があるモデルを提案し、実験によりモデルの正当性を確認した [井川 90] 。

以下、簡単に相互作用モデルについて述べる。相互作用モデルにおいては、人間も機械もテンポを決定するための情報は人間のテンポ (ΔT_k) と機械のテンポ (Δt_k) のみである。ただし、人間のテンポ決定時には「曲調 (テンポ) を変えたい」という「意志」が働き、これを f_k と表す。このことから人間系のモデルは、

$$\Delta T_{n+1} = \sum_{k=1}^{u_1} a_k \Delta T_{n+1-k} + \sum_{k=1}^{u_2} b_k \Delta t_{n+1-k} + \sum_{k=1}^{u_3} f_{n+1-k}$$

機械系のモデルは、

$$\Delta t_{n+1} = \sum_{k=1}^{v_1} c_k \Delta T_{n+1-k} + \sum_{k=1}^{v_2} d_k \Delta t_{n+1-k}$$

と表すのが適当であるとしている。階数、係数は曲の拍子などに依存する。

このようにして、人間－機械の相互作用を考慮したモデルを提案し、その解析結果に基づき新しい実時間適応自動伴奏システムを構築した。

その後 1992 年に澤田らは、人間の意志としてステップ関数を用いて、人間と機械の相互作用を調べる実験も行なっている [澤田 92] 。

歌声自動伴奏システム

上で述べたように、大照研究室ではすでに電子楽器の演奏に対する自動伴奏システムについて報告している [直井 89, 直井 90, 井川 90] 。その後、井上らは人の歌声に対して自動伴奏をすることを試みている [井上 92b] 。

この研究ではマイクからの入力信号を A/D 変換し、このデータを DSP で処理、歌声のピッチ検出および音高の決定を行なう。そのとき 3 周期の平均周波数にもっとも近い音高を歌声の音高として決定する。この音高を MIDI 信号として出力し、以前開発された自動伴奏システム [井川 90] で伴奏を出力した。

次に井上らはこのシステムを拡張し、任意の調で、また曲の途中から歌っても、歌唱の進行している小節を判定し、自動的に伴奏をつけるシステムを開発した [井上 92a] 。

このシステムでは歌声に対して決定される音高の時間的変化パターンを見ることによって楽譜上の位置を判定する。つまり歌声の変化パターンとあらかじめ持っている楽譜データから得られる変化パターンのマッチングを取ることによって、どこを歌っているのかを知る。このとき歌声のピッチの絶対値ではなく、時間的な変化パターンによって、楽譜上の位置を判定しているので正確に原曲の調で歌う必要はない。また曲の途中から任意の調で歌い出しても、楽譜上の演奏位置を判定し自動的に伴奏をつけることができる。この場合は、あらかじめ途中の特徴的なメロディ部分をいくつか与えておき、これらの変化パターンと歌声の時間的変化パターンとを比較し、最も近いパターンを歌われているところと判断し、その演奏位置、演奏テンポ、調に合わせて伴奏を出力する。

4.3.7 大阪大学（井口研究室）での研究

大阪大学の竹内らは、伴奏システムが独奏者の追跡に失敗した時の処理を提案し、また音楽的な「間」への対応を考えている [竹内 91] 。

リアルタイム性を考慮した自動伴奏システム

システムは常にソリストが現在楽譜上のどの位置を演奏しているかを追跡しており、楽譜と違う音符（音高）が入力され、どこを演奏しているのか分からなくなると、楽譜の他の位置に一致する部分があるかどうかを探す。一致する部分があればそこから自動伴奏を続行する。なければ楽譜上のどこかと一致するような旋律が入力されるまで伴奏を止めたまま待ち続ける。これにより、ミスタッチや曲の途中が飛ばされた場合にも正しく追従することができる。

システムには発想記号の情報もあらかじめ入力されており、伴奏中に発想記号の付けられた音符に出会うと、その発想記号に対するルールを適用してタイムテーブルを作成する。また演奏中に「間」が挿入された場合、システムが「間」というものを知識として持っていないと、その部分はテンポの変化と判断されてしまう。この問題を解消するため、現段階では「間」が挿入されるであろう位置にあらかじめフラグを立てておき、音符が楽譜に記述された本来の長さよりも長く演奏された場合に、それが「間」であるのかそれともテンポの変化であるのかを判別し適切な伴奏を実現している。

4.3.8 東京工業大学（田中研究室）での研究

東京工業大学の堀内らは、音楽の状況に応じて伴奏システムの独奏者に対する追従の度合を変化させ、独奏者をリードしたり、あるいは逆に独奏者に合わせて伴奏する方法を提案した [堀内 92a, 堀内 92b] 。

伴奏者の自主性を考慮した自動伴奏システム

従来の伴奏システムの研究では、独奏者にぴったりと合わせた演奏がシステムの目標であった。しかし、独奏者と伴奏者による演奏の本来の目的は良いアンサンブルを行なうことであって、必ずしもぴったりと合わせる演奏ではない。良いアンサンブルを行なうためには、独奏者の優れた音楽性はもちろんのこと、伴奏者も優れた音楽性を持ち、独奏者に合わせるのと同時に、自主性を持って優れた音楽表現を行なわなければならない。

この研究では自動伴奏システムにおいて伴奏者の自主性を考慮することにより、自然で人間的な（機械的でない）演奏が可能となる自動伴奏システムを新たに提案している [堀内 92a] [堀内 92b]。そのとき伴奏システムは内部に独自の演奏プランを持ち、自主的な演奏を行ないながら独奏者に合わせて伴奏パートの演奏を行なう。ここで収束時間という考え方を導入する。収束時間とは伴奏システムのテンポを修正して独奏者と演奏している位置が等しくなると考えられる時刻までにかかる時間と定義する。この収束時間の考えを用いて独奏者の演奏に合わせることにより、伴奏システムは（自分の演奏プランに基づく）自主的な演奏を行ないながら自然な変化で独奏者と合わせた伴奏が可能となる。また従来のような機械主導型のアンサンブルや人間主導型のアンサンブルではなく、状況に応じてそれらのアンサンブルの形態を変化させることにより、お互いのバランスのとれた良いアンサンブルを可能にする。そこで伴奏者の演奏を独奏者の演奏に比べてどの程度独立して演奏させるかという尺度として伴奏者の独奏者に対する依存度を導入する。この依存度にしたがって伴奏システムの独奏者に対する追従性を変化させる。すなわち独奏者がアンサンブルの中心となり、その主導権を握るような状況では依存度が高く、伴奏システムは独奏者の演奏に十分に合わせて演奏しなければならない。そこで依存度が高いほど収束時間を初期値よりも短くし、素早く独奏者の演奏に追従できるようにする。逆に伴奏者がアンサンブルの中心となり、その主導権を握るような状況では、依存度が低く、伴奏システムは独奏者の演奏にそれほど影響されずに自主的な演奏を行なわねばならない。そこで依存度に比例して収束時間を長くしてシステムの演奏プランに基づくシステムなりの自主的な演奏を行なうようにする。

4.3.9 おわりに

従来の研究では、独奏者の演奏入力にアコースティック音源を用いることは困難であり、多くの研究ではその入力を MIDI 楽器に限定していた。そのため初期の研究ではリアルタイム・ピッチ抽出を行なうために、例えばフルートのキーの上に光学センサーを置いてピッチ抽出の助けとしていた [Ver84]。しかし DSP など、音響処理のハードウェア、ソフトウェアの発展により、最近ではリアルタイム・ピッチ抽出が可能となっている。そのため今後は人間の声やアコースティック楽器の伴奏が中心となるであろう。

テンポ・トラッキングの研究は、独奏者の楽譜情報をあらかじめ与えずに、正確に認識することが今後の研究課題となるであろう。

並列処理や複数の人間の演奏を入力とした協調型演奏システムなどの研究も今後行なわれるであろう。

また、従来は独奏者から伴奏システムへの方向の情報の流れが重視されていたが、実際の演奏現場では、お互いがインタラクティブに演奏を行なうため、伴奏システムから独奏者への逆方向の作用（反作用）も考慮すべきであると考えられる。伴奏システムは人間の独奏者とアンサンブルを行なうシステムであるので、良いアンサンブルを行なうためにはこのことは非常に重要となるであろう。

参考文献

[AD90] Paul E. Allen and Roger B. Dannenberg. Tracking Musical Beats in Real Time. In *Proceedings of International Computer Music Conference*, pp. 140–143, 1990.

[Bai91] Bridget Baird. The Artificially Intelligent Computer Performer and Parallel Processing. In *Proceedings of International Computer Music Conference*, pp. 340–343, 1991.

[BBZ89] Bridget Baird, Donald Blevins, and Noel Zahler. The Artificially Intelligent Computer Performer on the Macintosh II and a Pattern Matching Algorithm for Real-time Interactive Performance. In *Proceedings of International Computer Music Conference*, pp. 13–16, 1989.

[BD85] Joshua J. Bloch and Roger B. Dannenberg. Real-Time Computer Accompaniment of Keyboard Performances. In *Proceedings of International Computer Music Conference*, pp. 279–289, 1985.

[BMS89] Bob Boie, Max Mathews, and Andy Schloss. The radio drum as a synthesizer controller. In *Proceedings of 1989 ICMC*, pp. 42–45, 1989.

[CGN91] Joe Chung, Neil Gershenfeld, and Marry Ann Norris. A development environment for string hyperinstrument. In *Proceedings of 1991 ICMC*, pp. 150–152, 1991.

[Dan84] Roger B. Dannenberg. An On-Line Algorithm for Real-Time Accompaniment. In *Proceedings of International Computer Music Conference*, pp. 193–198, 1984.

[DM88] Roger B. Dannenberg and Hirofumi Mukaino. New Techniques for Enhanced Quality of Computer Accompaniment. In *Proceedings of International Computer Music Conference*, pp. 243–249, 1988.

[DMR87] R. B. Dannenberg and B. Mont-Reynaud. Following an Improvisation in Real-Time. In *Proceedings of International Computer Music Conference*, pp. 241–248, 1987.

[Dri91] Anthonie Driesse. Real-Time Tempo Tracking Using Rules to Analyze Rhythmic Qualities. In *Proceedings of International Computer Music Conference*, pp. 578–581, 1991.

[FWZ91] Adrian Freed, David Wessel, and David Zicarelli. Max objects for media integration. In *Proceedings of 1991 ICMC*, pp. 397–400, 1991.

[KG89] David Keane and Peter Gross. The midi baton. In *Proceedings of 1989 ICMC*, pp. 151–154, 1989.

[KW91] David Keane and Kevin Wood. The midi baton iii. In *Proceedings of 1991 ICMC*, pp. 541–544, 1991.

[LFW91] Michael Lee, Adrian Freed, and David Wessel. Real-time neural network processing of gestural and acoustic signals. In *Proceedings of 1991 ICMC*, pp. 277–280, 1991.

[Loh86] Henning Lohner. The upic sytem: A user's report. *CMJ*, Vol. 10, No. 4, pp. 42–49, winter 1986.

[Mat89] Max Mathews. The conductor program and mechanical baton. In *Proceedings of 1989 International Symposium on Music and Information Science*, pp. 58–70, 1989.

[MC89] Tod Machover and Joe Chung. Hyperinstrument:musically intelligent and interactive performance and creativity systems. In *Proceedings of 1989 ICMC*, pp. 186–190, 1989.

[MRS90] Gerard Marino, Jean-Michel Raczinski, and Marie-Helene Serra. The new upic system. In *Proceedings of 1990 ICMC*, pp. 249–252, 1990.

[MRS91] Gerard Marino, Jean-Michel Raczinski, and Marie-Helene Serra. New upic system demonstration. In *Proceedings of 1991 ICMC*, pp. 567–570, 1991.

[PL91] Bruce Pennycook and Chris Lea. T-max : a parallel processing development system for max. In *Proceedings of 1991 ICMC*, pp. 229–233, 1991.

[PZ90] Miller Puckette and David Zicarelli. *MAX Development Package*. Opcode Systems, 1990.

[RM88] Dean Rubine and Paul McAvinney. The videoharp. In *Proceedings of 1988 ICMC*, pp. 49–55, 1988.

[RM90] Dean Rubine and Paul McAvinney. Programmable fingertracking instrument controllers. *CMJ*, Vol. 14, No. 1, pp. 26–41, Spring 1990.

[Ver84] Barry Vercoe. The Synthetic Performer in the Context of Live Performance. In *Proceedings of International Computer Music Conference*, pp. 199–200, 1984.

[VP85] Barry Vercoe and Miller Puckette. Synthetic Rehearsal: Training the Synthetic Performer. In *Proceedings of International Computer Music Conference*, pp. 275–278, 1985.

[井上92a] 井上渉, 橋本周司, 大照完. 歌声自動伴奏システム. 音楽情報科学研究会 夏のシンポジウム '92 Paper Session Proceedings, pp. 79–82, 1992.

[井上92b] 井上渉, 磯貝昌幸, 橋本周司, 大照完. 歌声のピッチ検出による自動伴奏システム. 情報処理学会 第 44 回 全国大会 講演論文集, pp. 1.387–1.388, 1992.

[井川90] 井川孝之, 直井邦彰, 大照完, 橋本周司. 相互作用モデルによる実時間適応自動伴奏とその動作解析. 電子情報通信学会 1990 年 全国大会 講演論文集, pp. 7–216–7–216, 1990.

[大照 88] 大照完, ほか. 人間のリズムを考慮した演奏情報から楽譜情報への一変換法. 電子情報通信学会 1988 年 春季全国大会 講演論文集, pp. D–414, 1988.

[竹内 91] 竹内庸博, 才脇直樹, 井口征士. リアルタイム性を考慮した自動伴奏システム. 音楽情報科学研究会会報, Vol. 38, pp. 24–25, August 1991. 夏のシンポジウム '91.

[直井 89] 直井邦彰, 大照完, 橋本周司. 実時間拍検出機能を用いた自動伴奏システム. 日本音響学会 講演論文集, pp. 465–466, March 1989.

[直井 90] 直井邦彰, 大照完, 橋本周司. 演奏情報処理の新しい一つの試み. 電子情報通信学会 1990 年 全国大会 講演論文集, pp. 7–215–7–215, 1990.

[堀内 92a] 堀内靖雄, スパワリークリッサダー, 田中穂積. 伴奏者の自主性を考慮した自動伴奏システム. 情報処理学会 第 44 回 全国大会 講演論文集, pp. 1.405–1.406, 1992.

[堀内 92b] 堀内靖雄, 藤井敦, 田中穂積. 伴奏者の自主性を考慮した自動伴奏システム. 音楽情報科学研究会 夏のシンポジウム '92 Paper Session Proceedings, pp. 73–78, 1992.

[澤田 92] 澤田秀之, 磯貝昌幸, 橋本周司, 大照完. 音楽演奏における人間と機械の協調動作について. 情報処理学会 第 44 回 全国大会 講演論文集, pp. 1.389–1.390, 1992.

第 5 章

音楽情報科学に関する諸分野

この章では、音楽情報科学の関連領域として以下について解説する。はじめに、前章まで解説してきた技術を背景に、その結実として作られてきたコンピュータ音楽の制作環境と歴史について述べる。続いて、音楽学に計算機科学がどのように応用され得るか概説する。最後に、今後の新しい分野と期待される音楽と脳波信号処理研究について解説する。

5.1　コンピュータ音楽の制作環境　(嶋津)

5.1.1　音楽とテクノロジーの距離

今日、多くのテクノロジーの成果を無視しては音楽を語ることができないだろう。「冷たい表現」[1]を得て、はじめて見えてくる世界を、音楽はこれまで、有効に取り入れてきたのである。

コンピュータ音楽は、テクノロジーの音楽への利用の具体的でまた象徴的でもある成果の一つと考えることができる。そしてまたそれは、音楽情報科学の芸術との共有部分でもある。しかし、今日、我々を取り巻く環境は、その実践の場も含めて、それまでに経てきた発展的成果を十分には得ていないように思われる。

その理由に、わが国においては、多くの芸術家が彼らの作品のテクノロジーへの応用に関心がないのと同じように、多くの科学者、技術者の関心が芸術にまで及ばないばかりか、彼らの友人に芸術家をあまり持たず、自身、芸術的展開への応用に関心がないということを挙げることができるかもしれない。しかし実際には、お互い関心があっても、それらを有効に結び付けうる社会上のシステムが欠落していると言ったほうがより正確なのではなかろうか。今日、強く結びつきあっているテクノロジカルな研究とその応用となる芸術活動は、残念ながら日本ではあまり強い関連性を持っていないと言うこともできよう。

そこで、そうした側面を明確にするために、コンピュータ音楽をその直接的先輩である電子音楽と連続させて、欧米におけるこれらの音楽の展開と日本における展開とを比較してみる。

5.1.2　コンピュータ音楽の歴史的概要

欧米各国が辿ってきたこのコンピュータ音楽の歴史をもう一度さかのぼってみる[2]。コンピュータ音楽の成立をどこにみるかは、コンピュータ音楽をどう考えるかによって異なるだろう。コンピュータについて考えるとき、そのハードとソフトの両面を考えてみる必要があるように、コンピュータ音楽を、コンピュータと音楽、言い替えてみれば、テクノロジーと音楽の両面から、これらを論じてみることもできるだろうし、テクノロジーが先か音楽が先かということになれば、

[1]嶋津武仁著「冷たい表現・・・振動する音楽」(季刊誌『MUSIC TODAY』 1990)、 p33

[2]嶋津武仁著「音楽とテクノロジー (1)」(福島大学教育学部論集第 38 号 1985 年) 及び「音楽創造とコンピュータ - 音楽とテクノロジー (2)-」(福島大学教育学部論集第 39 号 1986 年) 参照

歴史的にはほとんど平行して展開してきているといっていいだろう。また、時にそれは一体なものとなって展開してきたものでもあるといってもいいだろう。直接間接に関わらず、関わりかたの強さの多少の差はあっても、個々の事象によってその成立を語るより、様々な側面で、多角的なアプローチがなされ、その土壌や、背景、環境ができあがってきたものであり、それが今日のコンピュータ音楽の持つ様々な側面を形成していると考えるべきであろう。

コンピュータ音楽への予感　コンピュータの直接的開発が始まる以前に音楽の歴史の中に、コンピュータ音楽の予感をさせる多くの事例をみることができることであろう。それが、今日の環境を構成する不可欠な要因になっていると考えることができる。今世紀以前に起こったことを挙げてみると以下のようなものがある。

- オルガン (機関) の発明 (紀元前?)
- 永久機関、永遠運動への夢と試み
- アタナシウス・キルヒャーの作曲のための機械装置へのアイディア (17 世紀)
- オルゴールあるいはミュージカル・ボックスと呼ばれた音具 (18 世紀)
- 鍵盤楽器を含む、様々な西欧の楽器の工夫と完成 (18ー 19 世紀) [3]
- 「自動楽器」 (オートマトン automaton) への試みと自動ピアノ (19 世紀) の製作
- 電気の発見と実用
- 音楽への表現へのノイズ、楽器以外の手段の取り込み

電子音楽の展開　今世紀に入って電気楽器の発明が続き、それと平行して、「宣言」の時代と呼んでも良いほどの芸術への新しい考え方を提起する様々な動きが現れた[4]。それはそれまでの西欧中心の文化や表現方法、あるいは表現のシステムへの疑問から引き起こされたものと言っていいだろうし、それまでの音楽芸術への見方とは異なる、新しい視角の変化によるものと言ってもいいだろう。未来派、ダダイズム、シュールレアリズム等々。

ルイジ・ルッソロの「騒音音楽」は第 2 次大戦後のピエール・シェフェールの直接的先駆だけではなく、その「ブラックボックス」的楽器”イントナルモーリ”によって、その後の世界を暗示しているように思える。

ダダは、芸術の考え方への変革をもたらし、領域を越えて表現芸術が交錯する原点となった。サティはその風のなかで、多くの今日的話題を提供している。ヴァーレーズは積極的に科学の考え方を音楽に取り込んだ。そしてその息子たちの中からジョン・ケージがその主張を展開する。「電子音楽」という言葉は第 2 次大戦後に生まれた言葉だが、この先輩達はもうすでにその歩を進めてきていた。

もう一つの進んだ歩 (あゆみ) は「電気楽器」の発明に見られる。

- Dynamophone(1902)
- Teremin(1924)
- Spharophon(1926)
- Dynaphone(1927)
- Ondes Martenot(1928)
- Electronde(1929)
- Trautonium(1930)

[3]“広い意味でのマシン音楽は、実は音楽史の最初からある。いわゆる器楽がそれだ。”足立美比古「マン and/or マシン」(『即興する音楽マシン』のプログラムへの寄稿、1988) より

[4]「未来派」や「ダダ」などの芸術の主張は、その前後のものと区別するために、「宣言」という方法が取られた。いわゆる「ロシア・アヴァンギャルド」においても受け継がれている。「共産党宣言」もそういう時代になされた。

- Hammond Organ(1934)

等々。シンセサイザーが生まれる30年ほど前である。

第2次大戦後にさらに多くの環境が整えられる。パリ、ラジオ・フランスの一角で、「ミュージック・コンクレート」 musique concrète の研究と制作が行われる (1948)。ヘルベルト・アイメルト Herbert Eimert を中心にドイツのケルン放送局のスタジオで「電子音楽」が生まれた。これらのスタジオでは、それまで物理学の世界で使われ、戦争によってその精度を増した様々な音響機器や技術 (テクノロジー) が使われる。その中でも最も重要な表現手段に組み込まれたのがテープ・レコーダ (録音機) である。

最初のシンセサイザ (RCA Sound Synthesizer) はオルソン Harry F.Olson によってコロンビア・プリンストン大学で RCA と共同開発された (1955)。その後モーグ Dr. R.A.Moog によるシンセも生まれ (1964)、主にポピュラー音楽で多く使われたが、充実していく過程で、ほとんど過去の「電子音楽スタジオ」の能力を1つのシステムに完備させていくことに成功していく。

テープを中心とした音楽表現によって、必然的にそれまで直感に頼っていた音楽の様々なパラメータを正確な量や数値で表現することになった。そしてこの電子音楽もそれ故に「テープ音楽」と呼ばれることになる。しかし、他方その多様な表現の追求の中で、「再生」性をカバーし、視覚性を強めて「ライブ・エレクトロニック・ミュージック」に展開していく。

コンピュータ音楽の展開　ノイマン Von Neumann による真空管を用いたコンピュータ ENIAC も戦後間もなく生まれる (1946)。コンピュータが音楽に用いられ過程をおおよそ十年の単位で概略してみると以下のようになるだろう;

(1) 40–50 年代: 過去の音楽スタイルや音楽作品のシミュレーションと分析
(2) 60 年代: 計算結果の作曲への応用と音楽プログラムの開発
(3) 70 年代: コンピュータ音楽センターの設立による、科学的成果の音楽への応用のシステム化
(4) 80 年代: パーソナル・コンピュータを組み込んだ機器の小型化と安価で優れた民生機器の販売によって、リアルタイムな演奏表現の拡大と、コンピュータ音楽の大衆化。
(5) 90 年代: まだ半分も来ていないので、想像の域ではあるが、音楽情報のネットワーク化とコンピュータ音楽技術のグローバルな均質化と音響空間の仮想現実性の強化。 (?)

(1) の例としては、ピアス、シャノンの乱数を用いた作曲シミュレーション (1949) が挙げられる。そのほか、オルソン、ベラーによるフォスターの歌曲の統計学的分析に基づくシミュレーション (1950)。ケープリンのモーツァルトの「音楽のさいころ遊び」のプログラミング (1955)。ピンカートンの子守歌のサンプルの分析による単旋律の作曲プログラムの作成 (1956) 等々。しかしこの時代を最も代表することになるのは、イリノイ大学のヒラーとアイザックソンによる様々な実験と、その成果としての「イリアック組曲」 (1956/58) であろう。この「作品」とそのための研究過程において、彼らは音楽における新しい出来事 (event) が、それまで形成された出来事 (主に「旋律」) によって制限される「マルコフ連鎖」をその「ランダム・プロセス」 (任意の数値の連続的発生) に用いた。しかし音楽的には単純な、主に過去の音楽データの利用に基づいたものであった。そして 50 年代中頃にはコンピュータの助け (AID) を得た作曲が楽譜で表現されはじめていたのである。それらはやはり「ランダム・プロセス」を利用したものであり[5]、次の時代もその成果を土台に為されていくことになる。

(2) の時代において、音楽の創造的作品にコンピュータが利用され、また音楽の為のインターフェースが作成され、それらはすでに完成の域にあったシンセサイザとつなげられた。クセナキス I. Xenakis はその「ストカスティック (推計) 音楽理論」によって、IBM のコンピュータに

[5] "Composition with Computer", Ch.Dodge and Th.A.Jerse,'Computer Music', Schirmer Books,1985

よる計算結果を器楽作品に応用して “ST/10-1,080262” や “ST/48-1,240162” といった作品を作曲した (1962)[6]。彼はその作品制作のためのコンピュータ・プログラム SMP (Stochastic Music Program) をつくり、それによって楽譜記述の形で器楽作品に応用した[7]。

「ベル研究所」のマックス・マシューズ M.v. Mathews の様々な試みもこの時代にはじまる。60 年代初め、彼は “Music 3” を作り、これはその後、プリンストン大学の ハウ H.S. Howe やウインハム G. Winham の “Music4” にそして、バーコウ Barry L. Vercoe の “Music360” や “Music11” に発展し、受け継がれていく。

マシューズはまた画期的コンピュータ音楽システム「グルーブ」 GROOVE を考案した (1968/69)。これはデジタル信号によってアナログ機器をコントロールする「ハイブリッド・システム」と呼ばれた。この「システム」はまたケルンの放送局スタジオからユトレヒト大学のソノロジー研究所に移ったケーニッヒ G.M.Koenig も取り入れ、「プロジェクト 1」および、「2」というシステムを形成した。彼のこのシステムを用いる作曲家が続き、そしてこの研究所は当時のヨーロッパにおける最も進んだコンピュータ音楽のセンターとなった。ケージは、ヒラーの協力を得て、コンピュータによる作品 “HPSCHD”[8]を作っている (1969)。同じ年に、リセ J.Ch.Risset は、コンピュータによる音響合成の結果を取り込んでサウンド・カタログを作成し、それに基づいた作品「ミューテイション 1」 Mutation I を作曲した。

(3) コンピュータを用いた音響合成や音響解析の技術でえられた成果は音楽制作において用いられる様々なデジタル技術を生み出した。チョウニング John Chowning は自ら開発した FM 音響合成技術によって、基本的にはキャリアとモジュレータという 2 つの素子を用いて、単純なモデルから複雑な音響を合成することに成功し、それを利用した「トュレナス」 Turenas(1972) を続けて ”Stria”(1976) などの作品を制作したが、テクノロジーに裏付けられながらも、その音楽的内容においても説得力をもつ音楽作品となっている。また同時にヨーロッパで放送局を中心に開設された「電子音楽スタジオ」から発展したマサチューセッツ工科大学 (MIT 「ミット」) やスタンフォード大学 (CCRMA)、イリノイ大学、カリフォルニア大学サンディエゴ校 (UCSD) といった大学の内部あるいは隣接機関にコンピュータ音楽の研究所ができ、更にはベル音響研究所や IRCAM といった専用のコンピュータ・ミュージック・センターが設立されるなど、高度なテクノロジーの成果を直接、創造的行為に結び付ける環境が整備された。そして、コンピュータ音楽の成果を交換し、公表する「コンピュータ音楽協会」 (ICMA) とその国際会議 (ICMC) が始まった (1974)。また「コンピュータ・ミュージック・ジャーナル」も発刊され (1976)、今日まで、この分野における、最も重要な雑誌になっている。

(4) 70 年代終わりからミニコンのレベルの処理機能を備えたコンピュータ・シンセサイザが生み出された。代表的なものとして、「フェアライト」 Fairlight CMI (Fairlight Instruments 社製) や「シンクラヴィーア」 Synclavier (New England Digital Corporation 社製) などがある。これらは高価な点では、アナログ・シンセサイザーのいきついたところ (モーグの「システム 55」など) に類似していた。しかしその一方、80 年代は、 MIDI 機器を代表とする様々な民生器によって、研究所から、勉強机にその「仕事場」が拡張され、一般の愛好家に至るまで「デジタル機器」を所有し、かれらの趣味に供されるものとなってくる。

(5) 80 年代から 90 年代にかけて、いわゆるメイン・フレームのコンピュータによる音楽のシステム研究は、ワークステーションやパーソナル・コンピュータに連動したシステムやソフト・ウエアに応用され、 MIDI による様々なボード (SMPTE/MIDI カードなど) や、 IRCAM、CCRMA などで開発されたソフト「マックス」 MaX、更にその機能を DSP (Digital Signal Pro-

[6]ST は Stochastic を、 /10 は編成 (楽器の数)、それ以降の数値は、コンピュータによる計算結果を出した日という。

[7]Xenakis, Iannis. “Formalized Music”, Bloomington,Indiana Univ.Press, 1971, p.182

[8]ハープシコードの子音

cessing) にまで拡大される、コンピュータ音楽システムのユニバーサルな展開が行われている。それまで各研究所が独自に研究開発してきたコンピュータ音楽は、その交流を深め、人的交流も盛んになって、いわばそうした情報と技術の「ネットワーク化」が進行しつつある。

5.1.3 わが国における展開

こうした欧米のコンピュータ音楽における展開は、テクノロジーが音楽に組み込まれて来た試行錯誤の歴史であり、音楽家と科学者のサイドからのお互いの接近への忍耐や努力、更には情熱の結果といってもいいものであり、それが社会や政治そのものをも動かして、得られた「成果」といってもいいだろう。

そうした「成果」から、日本の音楽家や科学者が常に隔離、隔絶した動きをしてきたわけではない。時として欧米の流れにシンクロするような様々な動きを見ることができる。

20 年代、ヨーロッパにおいて盛んに創意工夫された「電気楽器」の影響は、わが国においては、宮城道雄が彼の発明した八十絃箏に電気増幅器をつけたり (1929)、石田一治が三味線の電気楽器「かん弦」をつくった (1931) ことなどに見られる。

そうした黎明期における例の他にも、欧米の動向と連続あるいは同期した多くの事例を見ることもできる。

日本の電子音楽　わが国における「電子音楽」から「コンピュータ音楽」への展開を年代的に概括すると、多少の批判を込めて、以下のようになるだろう。

- 50 年代:「ミュージック・コンクレート」と「電子音楽」の西欧からの導入と模倣の時代
- 60 年代: 実験と創造の時代。社会的認知。東京オリンピックから「大阪万博」へ。
- 70 年代: ライブ・エレクトロニックの隆盛と電子音楽の「衰退」の時代
- 80 年代: ポピュラー音楽における「シンセサイザー音楽」と「パソコン音楽」の隆盛

日本における最初の実験は、民間放送局「文化放送 JOQR」で放送された (1953)。作曲家、黛敏郎の帰国第 1 作「ミュージック・コンクレートの為の xyz」である[9]。これはわが国における初めての「ミュージック・コンクレート」作品になった。

その後、ケルンの「電子音楽スタジオ」の資料を得て、NHK に初めての「電子音楽スタジオ」ができた (1955)。そしてここがそれ以後日本におけるこの音楽の中心的機関になり、多くの作品が創られ、すぐに放送されて紹介されたのだった。そこで最初の電子音楽作品も黛による〈素数の比系列による正弦波の音楽〉〈素数の比系列による変調波の音楽〉〈矩形波と鋸歯状波によるインヴェンション〉と題された 3 楽章からなる「習作 I」がつくられたのである (1955) [10]。更に同じ年、柴田南雄は NHK の第一と第二放送を利用した「立体放送のためのミュージック・コンクレート」をその「電子音楽スタジオ」で制作した。

翌年 2 月には「実験工房」のメンバー[11]によって「ミュージック・コンクレートと電子音楽のためのオーディション」がひらかれる。そして武満徹の「ヴァーカリズム A・I」も新日本放送の委嘱でこの年に生まれた。そしてそれと競うように黛による「7 のバリエーション」が NHK の「電子音楽スタジオ」で制作されたのである[12]。この数年はわが国において、電子音楽における最も盛んな創作期となった。芥川也寸志、諸井誠、三善晃などが作品を制作している。

---

[9]『武満徹の音楽』(秋山邦晴編「年譜」より、日本ビクター 1966)、及び柴田南雄『日本の電子音楽の歴史と現状』(「トランソニック」 1974) などに記載。

[10]柴田前掲書には、「統一的な題名を持たず」とある。P.49

[11]滝口修造の命名により 1951 年設立。メンバーに武満徹、湯浅譲二、鈴木博義、園田高弘、秋山邦晴らがいる。

[12]秋山前掲書

放送という媒体による演奏形態の成功、そのステレオ放送 (立体放送) 化、映画におけるサウンド・エフェクト (SE) から映画音楽そのものでの利用など、様々なテクノロジーの発展とそれを取り入れた媒介によって、社会的認知を得易い状況が展開していたこともそれらの大きな要因になっていたと思われる。またこれらの作曲家が同時に現代の音楽のアクティブな側面の一つとして行っていたことも、その後 (あるいは現在) を考えるうえで重要なことと思われる。

1957 年に黛によって作曲された「葵の上」は伝統音楽と電子音楽という組み合わせによって、日本における電子音楽の創造の新しい可能性をも世界に投じた。彼の様々な梵鐘を集めた「カンパノロジー」(1959) は、彼の同年のオーケストラ作品「ねはん交響曲」に生かされた。これは、リゲティがそのいくつかのオーケストラ作品に、またシュトックハウゼンやブーレーズが電子音楽の成果を器楽の作品に展開してきていたことにも通じていた。

その後、60 年代に入って、湯浅譲二、高橋悠治、一柳慧、松平頼暁が電子音楽による作品に着手している。これらの作曲家の作品は、電子音楽が、器楽の作品と同じ〈手法〉に基づくのではなく、電子音楽独自の世界を追求したことが特徴として挙げられる一方、その「電子的プロセス」が多様に使われ、もはや「コンクレート」と「エレクトロニック」を区別する必要がなくなることにもなったと言うことができるだろう。今日、「ミュージック・コンクレート」も「電子音楽」の一部と理解するのが一般的な解釈と思えるが、その始まりはこのころにあったと言っていいだろう。その時期にあって、黛は「オリンピック・カンパノロジー」(1964) を「東京オリンピック」のメインスタジアムに鳴り響かせた[13]のである。この作品は先の「カンパノロジー」のアイディアに基づいているが、全国の由緒ある梵鐘とそれをシミュレートする電子音とからなるものであった。

こうした〈実験の時代〉が終わろうとしていた 60 年代中頃に、わが国における電子音楽が新たな展開を必要としてきていた。そんな時期、シュトックハウゼンが来日した (1966)。彼の来日によって、それまで「資料」と「作品」によって電子音楽のシステムを考えていたところに「生」の知識とより新しい装置や考え方がもたらされたのである。この年、湯浅譲二はホワイト・ノイズのみを素材に、フィルター効果を最大に生かした「イコン」を制作している。また、新しい方向として、アコースティックなコントロールや、マイクや回路を通じて得た音源に変調を加えることなどを主体として、ライヴ性を高めた「ライブ・エレクトロニック音楽」がつくられるようになった。一柳や松平頼暁、石井真木がそうした作品を残している。

70 年に開催された「大阪万博」はそれまでの電子音楽の成果の総決算的な意味で象徴的な事業になった。この時代に活躍していた多くの作曲家が彼らのそれまでの成果を「展示」した点ばかりではなく、その後の電子音楽に残したものを考えるうえでも「総決算的」であった。そして、テクノロジーが創造的芸術音楽作品に関わりえた、日本における最後の「万博」になった。電子音楽がテープによる再現音楽としての「テープ音楽」に移っていった時代、すでにその概念は壊され始めていたのである。

70 年代までは、欧米のこの音楽の流れとそれほど距離を感じなかったが、その後しばらく「万博」での「成功」の〈亡霊〉に縛られ始めたわが国の状況はそれとは大きな違いをみせていくことになる。欧米では、様々な大学にスタジオ、さらには「センター」が建設され、施設の中心も「テープ・レコーダ」から「コンピュータ」に移り、それまでの成果を取り込むように、技術者、科学者、音楽家の総合的研究と制作によって、音楽に「科学」のメス (解析機、合成システム) とメジャー (測定装置) を加えていったのに対し、わが国では、大学の研究室 (スタジオ) において、民生機を中心に、作曲家個人によるシステムにおいて、小規模に展開する。そうした中にコロンビア・プリンストン大学など海外で制作活動が中心の篠原真のほか、「音響デザインの会」における江崎健次郎や、端山貢明、和田則彦、また「デザインの会」にも参加していた住谷智など、

[13]柴田前掲書

東京学芸大学を中心とするグループ「TATA」がその活動を初め、東京芸大でも、南弘明らが制作を行っていた。筆者が電子音楽の制作を始めたのはまさしくそんな時期に当たっていた。

しかし、わが国の現実の状況はもはや「名人芸と熱意だけでは」[13]成り立たないと考えた多くの作曲家がこの領域の作品から「手を引き」、それまでの成果を着実に受け継ぐシステムと施設を持ち得ないまま、<衰退>の時期を迎えてていたのである。そうした中で、活性化したのは、シンセサイザーによる音楽の表現手段を持った「ポピュラー音楽」においてであった。それに功績のあったのは、富田勲であろう。モーグの制作したシンセサイザーを用いたワルター・カルロス(現在はウインディ・カルロス)による映画音楽や「スイッチド・オン・バッハ」を発展させた彼の様々なシンセサイザーによるクラッシック音楽の編曲は、シンセサイザーをポピュラーな楽器にし、その後市販された安価なシンセによって、それを個人のレベルで購入することを可能にし、窒息しつつあったスタジオから電子音楽をポピュラーな媒介に引き出すことになった。そして、それはまた伝統的電子音楽の世界にも個人のレベルでこの音楽の制作を可能にさせていった。その一方「NHK スタジオ」でも電子音楽制作の経験のない「作曲家」に<優先的>にそのスタジオをオープンしていくなど、欧米の流れとは異なる<道>を進み始めたと言えよう。いわば、わが国の<総アマチュア化>指向に組み込まれて進行していく。

日本のコンピュータ音楽　日本におけるコンピュータの音楽への利用は、「万博」においてすでに江崎が行っているが、本格的には 74 年の武田明倫の「パノラミック・ソノーレ」が最初とされている[14]。

80 年代に入って、徐々に安価になってきたコンピュータやシンセを組み合わせて、アナログ・シンセをデジタル信号でコントロールする、いわゆる「ハイ・ブリッド・システム」から、ヤマハの DX7 などのデジタル・シンセの登場によって、コンピュータを「パソコン」でコントロールするという方法で作曲を行う作曲家が現れた。しかし、あまり本格的な作品は 80 年代後半を待たざるをえなかったように思える。そうした中で、海外のコンピュータ音楽センターや最先端的スタジオで制作や研究を行ってきた作曲家たちが、加わりはじめてその〈息〉を吹き返すことになったと言えよう。海外で制作を続ける湯浅譲二や独自のパソコンによるライブ作品を展開する高橋悠治のような、初期電子音楽からの作曲家は希少な例であるが、そうしたものを除いて、90 年代になって、やっとその活動に日の光が与えられて来つつあるように思える。

90 年の 11 月に国立音楽大学にわが国初めての「コンピュータ音楽センター」が設立され、IRCAM や CCRMA (スタンフォード大学のコンピュータ音楽センター)、カリフォルニア大学サンディエゴ校 (UCSD) の「コンピュータ・オーディオ研究所」(CARL) から講師を呼んでその充実を計っている[15]。また、パーソナル・レベルでのコンピュータを取り込んだ作曲家たちの活動も今後に大きな可能性を見せている。その多くは「マック」[16]などを用いて、音楽ソフトを利用した音楽を作ることが主な方法である。そして、主に戦後世代の 30–40 代の作曲家達によって、この分野の盛んな活動が展開されてきている。

今年 (1993) 東京で ICMC (コンピュータ音楽国際会議) が開かれる。これまで欧米でのみ開かれていたこのコンフェレンスがわが国で開かれることで、欧米との同時代的交流が展開することが望まれている。

---

[14]武田「コンピュータ・ミュージック」(『新音楽辞典〈楽語〉』音楽の友社 1977) より

[15]「特集 6」(高橋信之『MacJapan(91-3)』 1991) より

[16]一連のマッキントッシュ製のパーソナル・コンピュータ

5.1.4 作曲環境上の課題

電子音楽における制作上の課題 非デジタルな制作過程における課題は、アナログ・テープ及びその録音機器による制作プロセスが中心である為に生じた以下の課題に対し、演奏形態、作品提示の方法で工夫がなされた。

- 「再生の音楽」(リアル性の欠如): ライブ・エレクトロニックや演奏家とのアンサンブルの演奏形態で補う。
- 「視覚性」の欠如:「舞踊」や「舞踏」、その他の「ダンス」、シアター・ピース、パフォーマンスや他の「視覚メディア」とのいわゆる「マルチ・メディア」作品として作曲する。
- 「テープ・ノイズ: Dolby, DBX などのノイズ・リダクションで対応。

更に「電子音」的操作の利点を取り込む以下のような方法の追求がなされた。例えば、

- 正確に計算されうる音楽データの使用: ピッチ (半音以下のマイクロ・チューニングや倍音、「和音」、「音響」)、音価 (音の長さ)、ダイナミックス、それらの組み合わせ。
- マルチ・ディメンション、マルチ・チャンネルによる、「音響」の移動性を含んだ、音響空間の構築。

などがある。

この音楽の制作現場である電子音楽スタジオもその制作環境上の重要な課題を提示している。個人的なものを除いて、理想的なスタジオのあり方としては、

- プロフェッショナルな音響機器の使用とその専門家の専門知識、専門技術の音楽アイディアへの取り込みを可能にする。
- この分野における音楽家の作品をその制作過程も含めて、参照できる環境の中で制作を行う。
- 作曲家が、そのスタジオに合わせるのではなく、スタジオが作曲家の要求に答えるように、拡充される、又そうしたフレキシビリティ (柔軟性) を持つ。
- 他のスタジオとの交流 (スタッフ、作曲家を含む) とデータの共有化。

といったことが求められた。

コンピュータ音楽における制作上の課題と可能性 「電子音楽」で生じた課題の解決はいうまでもないが、デジタル化によって生まれた新たなる問題にも対応することが求められて来た。例えば、使用するシステムのソフトウェアのほか、ハードに対する知識と技術を作曲者自らが体得しなければならない環境ではあるが、その反面、作曲家にとってプログラミングの経験は、音楽におけるそれまでの「楽譜」とは異なる別な「言語」で表現することの意味を考えさせ、音楽の表現の根源的意味にも通じる、様々なアイディアを作品に提供する可能性が広がっている。

パーソナル・コンピュータの制作過程への導入は、コンサート会場にそれを運び込み、リアル・タイムに音楽を創造する方法を可能にするばかりか、かっての「テープ」に〈コントロール〉されていた演奏家の〈息〉や〈間〉といった音楽的〈ゆれ〉や〈ゆらぎ〉といった演奏の不確定な要素を取り入れることを可能にしている。すなわち、マシン (コンピュータ) が演奏者に同期するという、人間の生きた表現をサポートするシステムの構築を可能にしてきている。また、それまで不確定であった様々な音楽表現上のパラメータの「再現性」を高めてきている。

結果として、マシンがマンにとって、その鏡となりうるような音楽づくりを可能にしてきている。

5.1.5　コンピュータ音楽の意義

用語としての「コンピュータ音楽」　同一の用語が、その使われる世界で意味を異にすることはよくある。様々な領域が重なりあっている言葉ならなおさらであろう。「コンピュータ音楽」という言葉にもそういった「同音異義」的な違いが見られる。同じような違いは「電子音楽」という言葉にもみられる。しかし、時として、〈芸術〉に所属すると自負する〈アーティスト〉の多くが、「ポピュラー音楽」の意味で用いたり、〈芸術〉(アート)そのものの意味もかなり異なりをみせてきているのであるから、益々その判断の拠り所を失ってきていると言えよう。作曲家が、そうした違いを「置き去り」にして、この音楽の世界で創造活動を行うことは危険であろう。

コンピュータ音楽の意味するもの　コンピュータ音楽は「コンピュータ」と「音楽」という2つの言葉が単純に結び付けられたものではなく、音楽がテクノロジーと結びついてきた結果として、音楽にとっては、その幅を広げさせ、その表現の精度を高めたばかりではなく、それまでは共有の部分を持たなかったあらゆる分野、領域との直接的結びつきの可能性をもたらす媒体になってきている。そしてそれが我々の生活に今までにない潤いと精神的豊かさを期待させている。芸術の手段としてばかりか、生活環境や治療にまでも展開され、広範な人間の活動、生活に関与しつつある。そうしたなか、コンピュータを用いて音楽創造を深めて行けば行くほど、その行為を通じて、音楽の持つ意味、それを聴く、あるいは創る人間の持つ意味を考えさせてくる。

コンピュータは、好む好まざるに関わらず、我々の周囲に存在する音楽に、新しかろうが古かろうが、様々な表現(演奏、作曲、鑑賞その他)のレベルでいやおうなく関わってきている。コンピュータによる表現に「機械的」、「非人間的」といったレッテルを貼ってしまって、無視したり、否定するための理由としようとするなら、そこからはおよそ何の成果ももたらさないであろう。しかし他方において、テクノロジーの発展が必ずしもそれを用いた音楽作品の音楽的質を高めるなどと考えるものがいるとしたら、同様にとんでもない誤解といっていいだろう。音楽の質、音楽性、音楽作品としての価値のために、時として、テクノロジーは何の役にも立っていないことも事実であろう。

最後に電子音楽及びコンピュータ音楽関係の参考文献を挙げて、本節の結びとする。

参考文献

[1] L. Hiller: “Informationtheorie und Computermusik, Darmstädter Beiträge zur Neuen Musik”, 1964 Mainz Schott

[2] H. Eimert, H.U.Humpert: “Das Lexikon der elektronischen Musik”, 1973, Gustav Bosse Verlag

[3] M.V. Mathews, J.R. Pierce: “Current Directions in Computer Music Research”, 1991 MIT Press

[4] H.F. オルソン (平岡訳): “音楽工学”, 1966 新光社

[5] H.B. リンカーン: “コンピュータと音楽”, 1972 カワイ楽譜

[6] 篠原 真: “電子音楽について”, トランソニック 4 秋号, 1974 全音楽譜

[7] A.H. ドイチュ (梯訳): “SYNTHESIS シンセサイザー その歴史と理論と実際”, 1979 パイパーズ (原文 1976)

[8] P. シェフェール (馬淵訳): “音楽言語・情報理論”、エピステーメ 8+9 月号, 1976 朝日出版

[9] 松平 頼暁: “20·5 世紀の音楽”, 1984 青土社

5.2 音楽学における計算機応用 (鈴木)

計算機を用いて音楽そのものを実践したり研究しようとした場合、二つのアプローチが考えられる。一つは計算機科学あるいは記号論や言語論を用いて、音楽そのものを記述することや音楽そのものを創ることである。これらに関しては、前述しているのでここでは触れない。

もう一つのアプローチは、人文系の学問分野である「音楽学」の研究に計算機科学を応用しようとするものである。ここでは、この音楽学における計算機応用がどのように行われてきて、今後どのように発展していくであろうか解説する。

この分野は、計算機を用いた作曲と同様に、計算機の歴史の比較的初期に研究が始められている。1960年代には、統計的手法を用いて音楽作品を定量的に評価しようとする試みが行われている。しかし、当時の計算機環境は現在と比べると、はるかに扱いにくいものであった。音楽学者がしたいことは、専門の計算機科学者にプログラミングしてもらわなくてはならず、必ずしも意志疎通がうまく行き、研究成果が上がったとは言えないようである。そういった停滞感もあって、以後この分野が計算機科学や技術の進歩に比例して発展してきたとは言えない。これらに関しては、本節の最後に参考文献を上げるに止めておくことにする。(注: ここでは主として日本国内の研究である)

元来、音楽学は扱う分野も幅広く、研究者個人によって目的やアプローチも多種多様である。これは他の人文系の学問 (例えば歴史学や文化人類学等) でも同様であろう。

音楽学と言った場合、今まで一般的にその意味するところは、いわゆる西洋音楽学であった。しかし、世界中には多種多様の民族音楽や大衆音楽 (ポピュラー音楽) が存在している。西洋対非西洋という枠組みで、音楽を考えるのでなく、あらゆる音楽を対象とした「音楽学」として研究すべきだという認識が広がってきている。そういった枠組みにおいて、何か科学的アプローチを用いて、研究が進められないか考えることは有意義であると思う。

一方、計算機科学技術は近年飛躍的な発展を見せていると言えよう。今までは、より速く、より安く、より便利なといった我々の生活に直接役に立つ分野で驚異的とも言える進歩をしてきた。そういった背景のもとに、より「柔らかい学問」である音楽学に計算機科学が貢献できるであろうことをもっと考えていいと思う。

そのような立場で、一つ例を考えてみる。

西洋音楽だけでなく、総ての民族の音楽を扱い、あるいは音楽を通して人類学的な研究をしようという分野に、いわゆる民族音楽学がある。(民族音楽学も、広い意味での音楽学であるべきであるという主張は、前述している。) この分野では、民族固有の文化的文脈から生まれた「音楽」を対象にすることになり、音楽そのものだけではなく、その社会の中での音楽という点からも、膨大な情報が必要になる。しかも、「音楽」という言葉の意味さえもが民族毎に異なるのであるから、その扱う資料の形式や質も各々異なる可能性がある。さらには、その「音楽」は歴史的、時間的な変容を繰り返している場合も多い。従って、そのような音楽を科学的に学問として扱うには、その方法論も含めて非常に難しいものがあるであろう。民族音楽研究者個人の直感 (あるいは鋭い洞察力) によって仮説が立てられ、それを実証するのに膨大な努力を要しているというのが現実ではなかろうか。そのように、なかなか全体像を把握しにくい民族音楽学に、何か計算機科学が貢献できないであろうか?

ここでは、一例として「民族音楽とデータベース」という視点から考えてみたい。

民族音楽学の重要な手法の一つとして、現地でのフィールドワークによるデータ収集と解析作業がある。その解析において、計算機科学が役に立てないであろうか? すなわち、大量にフィールドで集めたデータをデータベース化して、人間の直感では把握できないようなデータの大海の中から、計算機を使って音楽的な文脈を見つけ出すことが出来ないであろうか? データベースでは検索ができることだけでも有用であり、単にデータベース化することだけでも意義は十分にあ

るであろう。(もっともどのようなデータベースを構築するかは、データ入力の労力と共に十分に難しいテーマではあるが。)

ここではそこから一歩踏み込み、研究者 (ここでは民族音楽学者) が自由にデータベースに思い付いた問い合わせを発行し、それにデータベースシステムが答えてくれるシステムを作り出そうということに触れる。すなわち、どこかの民謡と似ている民謡が別のところにどれくらい存在しているか、あるいは、継承されている時代背景から時間的な変遷がわかる、そういったことである。他にも音階の特徴や楽器の使われ方など、広範囲に研究者が思い付いた問い合わせに答えてくれるシステムである。

一方、計算機科学側からデータベースを見てみよう。既に多くのデータベースが実用化されている。ここには、ビジネス用途で社会生活に役に立っているものから学術研究目的のデータベースまで多数存在する。

しかしながら、一般のデータベース (主としてビジネス系の) では、現在は定型業務が中心で、ユーザの自由な問い合わせに即座に応答することは難しくなっている。ところが、近年ようやく立ち上がってきた関係データベースでは、そういったことが可能になりつつある。関係データベースが高速にできるようになると、ユーザが ad hoc に問い合わせを発行し、非定型なこともデータベースでできる (あるいはやりたい) ようになって来ている。例えば、ある人のカードの支払い項目から将来その人が購入しそうなものを推理して関係するダイレクトメールを出す、あるいは、ある店の販売実績の推移から同じように次期販売戦略を提案するといったことである。(このようなアプローチをデータベースマイニングと呼ぶ。)

この考え方は、データベースの応用として音楽学 (だけではなく人文系一般の研究) にも役に立つのではないかと考える。

同じような考え方で、データベースを応用している例として、日本史や古典文学をデータベース化した例がある。文部省国文学研究資料館では、岩波書店刊の「日本古典文学体系」全 100 巻の本文、全 3000 万文字のデータベースを現在作成中である。これが完成すれば、ある語句の出現頻度や文中の位置、あるいは時間的歴史的変遷がわかり、新しい発見ができるのではないかと期待されている。

音楽の研究分野で、データベースが構築されたという例は、残念ながらまだ多くはないが、国立歴史民俗博物館で日本各地の民謡を 1 万曲以上調査して、データベース化しようという試みが報告されている [23]。(以上のデータベース化に関する文献は本節の最後を参照のこと)

問題点としては、以下のようなことが考えられるであろう。

現在は、データベースで扱える情報はキャラクタベース (テキストや数値) が主である。本来は音楽などは、テキスト情報だけではなく、音そのもの (音声情報) や、演奏している姿や楽器の形 (映像情報) などマルチメディアで扱えれば、格段に情報量が増すのは明らかである。このためには、計算機科学側からもマルチメディアデータベースといった新しい分野の研究、構築が必要であり、今後の大きな課題と言えるだろう。

もう一つ、大きな問題としては、このような学際領域の研究においては、異なった領域の研究者の共同研究という作業が重要になってくることである。近年、研究者の領域は、益々専門化する傾向にある。それらの専門領域を横断した学際領域に新しい学問領域が生まれる可能性がある。音楽学者と計算機科学者の連係、共同研究によって新しい分野が生まれることを期待したい。また、そのような研究体制を容易に作り出せる環境を整えることも重要であろう。

音楽学関連の参考文献

[1] 波多野誼余夫: 音楽への情報論的接近, 音楽学,Vol.14(1-2), pp.54-64,1968

[2] 音楽分析学会 (波多野誼余夫・三上文子・大久保京子・佐野光司・柴田南雄・徳丸吉彦・槌屋

治紀): 統計的音楽分析 (1)- ドイツの子供の歌, 野村良雄先生還暦記念論文集『音と思索』,pp.89-107, 音楽之友社,1969
[3] 川野洋: 音楽的コミュニケーションの情報分析, 音楽学, Vol.15(3),pp.1-21,1969
[4] 徳丸吉彦: 分析的研究と定量的測定, 音楽教育学研究ノート 6, 音楽教育研究,Vol.13(10),pp.114-119,1970.10
[5] 武田明倫, 奥藤早苗: 旋律構造の統計的分析 –6 種の <刈干切唄> を対象として –, 武蔵野音楽大学研究紀要 V,1971
[6] 武田明倫, 玉松京子: 音楽分析のための電子計算機プログラム MUSANA II について, 音楽学会第 21 回全国大会, 音楽学,Vol.17(1),pp.37-39,1971
[7] 武田明倫: 箏曲段物の統計的分析, 音楽学会第 22 回全国大会, 音楽学, Vol. 17(3), pp.203–205, 1971
[8] 佐藤幸平: シンタックスとしての和声理論, 音楽学, Vol.19, No.4, 1973
[9] リュウェ,N.(井上訳): 音楽学と言語学, 言語,Vol.2,No.5,1973(原文 1967)
[10] 佐藤幸平:「記述的和声理論」の試み, 音楽学,Vol.21,No.3,1975
[11] 徳丸吉彦: 日本民謡の統計的分析, 国立音楽大学大学院研究年報,1975
[12] 峰岸由紀: 音楽記号学の諸問題, 音楽学,Vol.23,No.2,pp.134-144,1977
[13] 武田明倫: 日本伝統音楽の分析的研究 - 箏曲段物を中心として -, 音楽学, Vol.25(1),pp.22-45,1979
[14] 神前尚生: 形式音楽理論の背景あるいは応用分野, 音楽学会第 31 回全国大会, 音楽学,Vol.26(2-3),pp.151-153,1980
[15] 山本順人, 藤井知昭: 音楽情報の計算機処理, 音楽学会第 32 回全国大会, 音楽学, Vol.27 (3), pp.227, 1981
[16] 山本順人: 計算機による民族音楽への接近, bit,Vol.13,No.11,pp.1531-1537,1981.10
[17] 坂崎紀: 音程進行による旋律の定量的分析, 音楽学,Vol.28(2),pp.101-117,1982
[18] 神前尚生: 形式言語論に基づく形式音楽論 (1) – 生成文法の音楽への適用の概観, ならびに <形式音楽> の定義, 音楽学, Vol.28 (2), pp.89–100, 1982
[19] 神前尚生: 形式言語論に基づく形式音楽論 (2) – 句構造文法と音楽の基本構造 –, 音楽学, Vol.31 (2), pp.108–124, 1985
[20] 坂崎紀: 計算機と音楽の接点, コンピュータと音楽 (西村恕彦, 音楽情報科学研究会), pp.84-90, 共立出版,1987.9
[21] 金城厚: 日本・アジア音楽の採譜と音楽分析, 岩波講座 日本の音楽・アジアの音楽 7 研究の方法, 岩波書店,1989.4

データベース関連の参考文献

[22] 山本順人: 民族音楽とデータベース, コンピュータと音楽 (西村恕彦, 音楽情報科学研究会), pp.105–111, 共立出版, 1987 年 9 月
[23] 金城厚: 音楽学におけるデータベース (日本民謡のデータベース化を中心に), 音楽情報科学研究会会報,No.23,1989.1
[24] 杉田繁治: 人文科学におけるマルチメディアデータベース, 情報処理,Vol.28,No.6,1987.6
[25] 打浪清一: フィールド調査データ処理におけるマルチメディアデータベース, 情報処理, Vol.28, No.6, 1987 6 月号
[26] 星野聰: 日本史データベース, 情報処理,Vol.33,No.10,1992.10

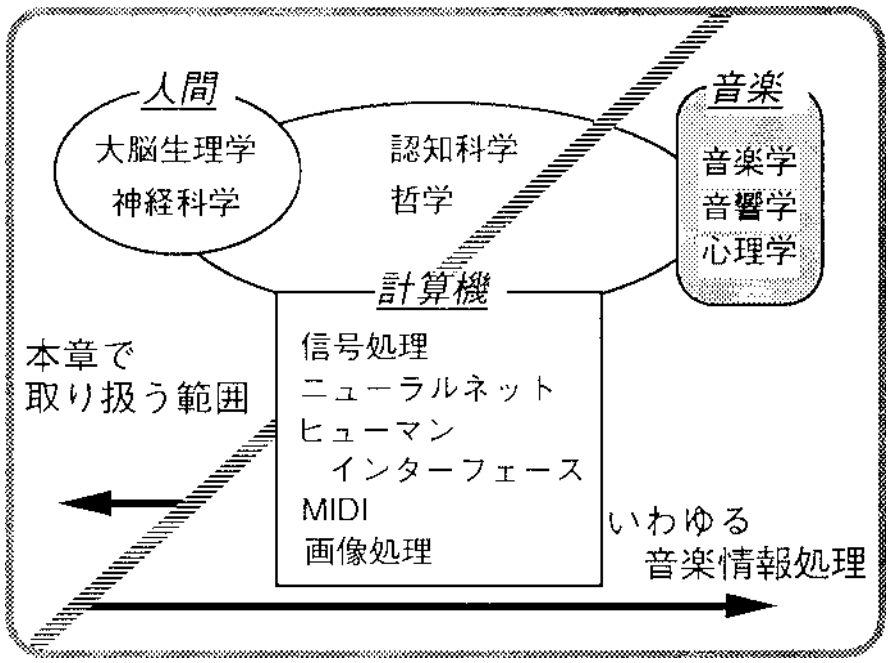


図 5.1: 音楽情報処理と脳研究の接点は・・・??

5.3 音楽と脳波信号処理 (才脇)

5.3.1 音楽と脳

これまでの各章で概観してきた音楽情報処理は, “音楽という対象を計算機上でいかに取り扱うのか” という視点からのアプローチであった. そして, 音楽のどのような側面に注目するのかによって, 画像処理や音響信号処理, あるいはヒューマンインターフェースといった, 工学的には異なる分野で開発されてきた要素技術を応用することができる [b1][b2][b3]. もっとも, 対象が音楽というカテゴリーで括られる以上, “音楽とは何であるのか” あるいは “計算機で扱える音楽はどのようなものであるのか” といった, 音楽そのものに対する視点を失う事はできない. その意味では, 音楽学や認知科学等の研究成果を視野に収めつつ, より本質的な「音楽に関する知識とその表現」を構築する必要がある [b4][b5][b6][b7][b8][b9][b10][b11][b12].

しかし, 普遍的な「音楽に関する知識とその表現」の体系は依然として見いだせていない [b13]. ここに, 計算機で音楽をいかに扱うかという問題と平行して, “音楽という対象を人間はどう処理しているのか”, あるいは逆に “人間がどのように処理しているものが音楽と呼ばれるのか” という視点から研究を進める必要性が生じる. このような研究は, 脳の方法論から学んだ工学的手法に基づいて人間の情報処理をシミュレートする方向と, 脳機能の解明そのものを目的とする方向に大別できる.

前者では, ニューラルネットワークを用いたセッションシステムやミンスキーの K-Line モデルを用いた作曲システムが良く知られている [b14][b15]. また後者の例では, スペリーに代表されるような左右脳に関する研究が有名で, 右脳が音楽脳と呼ばれるきっかけとなった [b16]. 今回, ニューラルネットワークに関しては別章が立てられているので, 本章では主に音楽と脳神経生理学の関連分野の研究に絞って概観する. そして, 脳活動解析の一手法である脳波解析のための要素技術, 特に脳波信号処理について簡単に紹介する (図 5.1).

脳の研究そのものは, 古代ギリシアの時代から行なわれていた. 中でも, プトレマイオス一世

時代のヘロフィスは, 当時世界で最も優れた研究機関であったアレクサンドリアのムセイオンにおける臨床観察から, 大脳が思考器官であり随意運動と外界認知にかかわっている事を報告している. しかし, 彼の研究が評価されるようになるのは近代を迎えてからであった. 19 世紀半ば頃, 失語症や言語障害と右半身の麻痺の相関が報告され, ようやく左右脳における機能差が認められるようになった. そして, 左半球と高次精神機能の関連性が盛んに研究されはじめた. 一方, 音楽と左右脳に関する当時の研究例は少ないが, 次のような非常に興味深い報告もなされている [b16]. 「57 歳のモーリス・ラベルは, 交通事故で左半球に損傷を受けた. 彼は音楽を鑑賞することや音の乱れや不協和音を聞き分けること, 自分が感じた音楽の喜びを書くことは出来た. しかし, 自分の頭のなかでイメージした音楽を作曲することはできなかった. ピアノを弾く事や, テンポを取って歌ったり, 音譜を読んだり書いたりすることもできなかった. 」(アラジュアーニ, 1948)

やがて, 1972 年にウィーンで開催された神経学に関するシンポジウムにおいて, 音楽の神経学が取り上げられると, ようやく「音楽と脳」の関係を学際的領域から追究する学問が認知されるようになった. 当時の研究成果は “Music and the Brain” に詳しい [b17]. しかし, 注目を集めていた音楽と左右脳の機能に関しては, 信頼のおける研究を総説した研究者のほとんどは, 音楽の機能をどちらか一方の半球に局在させるのは不可能である, と結論づけざるを得なかった [b16]. というのも, 音楽は音高や音長, 旋律, 音色, 和声, そしてリズムが様々に組み合わさって構成される上に, 音楽を聞く側にも初心者から高度に訓練された聞き手まで大きな幅が存在するためであった.

一方, 音楽ではなく単純な音刺激を用いて左右半球の優位性を調べる二音実験によって, 言語の左半球優位性と音の右半球優位性が示された. 角田は, DAF (Delayed Auditory Feedback) 効果を用いて独自の二音実験法を提案した [b18]. そして, 言語音や楽器音, 虫のなき声等自然音の半球優位性について実験を行い, 西洋人と日本人の脳における情報処理の違いを示した. さらに, 修得する言語の違いが脳の機能分化に影響を及ぼし, ひいては音に対する反応にも違いを生ずるようになる, という仮説を立てている. 彼によれば, 音楽は一般的に右脳で処理されるが, 西洋人が邦楽器の音や自然界の音も右脳で感じるのに対し, 日本人は左脳で意味づけをしながら聞いているという.

ところで最近では, 脳の機能が左右で異なるのみならず, 同一半球においてさえ異なった様々な機能が局在している事が明らかになり, 単純な左右脳の役割分担による情報処理のモデルでは説明不足である事が明白になりつつある. この延長線上に, 特定事象の記憶や認識に関与するニューロンが存在するという説がある. しかし, この考え方は人間の脳を観察して得られた知見というよりも, むしろニューラルネットワークで用いられる手法を脳に適用した仮説であると言える. 実際には, 脳は決まった機能を持ったパーツが集まって成り立っているのではなく, 脳の可塑性によって再組織され全体として機能する事により特異な能力を発揮しているように見受けられる [b19][b20].

これらの研究の究極の目的は, 心の働きを科学的に解明することである. そのためにミクロな神経細胞の働きや, それによって構成されている神経回路網の理解とともに, それらが全体として脳の高次機能へと構成される原理と機能の原理を知る必要がある. 半導体の知識のみから計算機の構成原理や動作原理を理解できないように, 要素レベルの知識のみでは脳は理解できない. 塚原は, 脳の研究におけるシステム的観点を重要視し, 脳を情報処理系とみなして高次機能を研究する情報神経生物学 (informational neurobiology) を提唱している [b19]. また, その背景には, 脳と心, 物質と精神, 分子と情報の対立といった, 現代科学に残された未知の領域としての「意識」の問題に対する積極的なアプローチが科学界で認められつつある, という事実がある [b21].

このような見地から脳の高次機能について研究するためには, 様々なメンタルタスクを与えた場合の反応を系統立てて分析する事が必要である. 一般的に, 分析対象として脳波や脳磁波が選

ばれる．その他, 酸素消費量で活動部位を特定できるPETや脳の形態そのものを調べることができるMRIも, これらの解析結果と組み合わされる事が多い．これらは, いずれも非侵襲で脳機能計測が可能という点で, 高次脳機能解明の手掛かりを提供する手段として欠かせない．対象とする健常人の脳組織に直接手を加えることは倫理的に見て許されることではないからである．中でも, もっとも一般的かつ古くから用いられている脳波による脳機能の計測は, 現在でも様々な形で研究が進められている．音楽に注目した例をあげれば, 音楽演奏中や傾聴中といった精神作業時における$Fm\theta$やα波を論じた研究例がある[b22][b23]．また, そうした脳波の特性を精神療法の指標として論じたものもある[b24]．さらに, 人間の可聴周波数帯域を越える周波数成分が, 人間の精神状態に影響を与えるという報告もある[b25].

このように, 音楽と人間の情緒活動との間に深い関係があるという研究の存在する一方, イメージ作用中に起こる大脳皮質の電気的活動には個人差が大きいことや, 同一人物においてもきわめて不規則であることが報告されている[b26]．従って, この分野の研究者は, 実験の際に与えるタスクとそのコントロールパラメータの設定に留意し, 詳細な実験データを公表する必要がある．また, 塚原が提案しているような情報神経生物学的モデルを構築し, 個人差や一過性の結果ではなく, モデルに一般性が存在する事を実験結果より証明しなければならない．また, 単純な音刺激によって誘発される電位について調べた研究の場合, 結果の客観性においては優れているものの, 直接高次精神活動を解析したことにはならない．現在, 低次の精神活動と高次の精神活動の関係についても明らかではないので, 双方の研究は平行して行なわれる必要がある．

以上でみてきたように,「音楽と脳」そのものはかなり古いテーマであるが, その関係を科学的に調べようという動きは比較的新しい．音楽と脳波に関して言えば, 様々な信号処理手法を用いて脳活動の有意な変化を解析しようという試みがなされ始めたという段階である[b27]．次節では, 音楽傾聴時の脳波を対象とした脳波信号処理法を, 才脇, 井口 (阪大基礎工), 井上 (阪大医学部), 片山, 堀 (岡山大医学部) らのグループの例をあげて紹介する[b28].

5.3.2 音楽傾聴時の脳波解析

脳波とは, 脳の活動や局所異常によって頭皮上に発生する$\pm 50\mu V$程度の電位変化で, 様々な周波数成分を含んでいる．音楽を聞いている際に, 何らかの形で心理的変化が生理的反応である脳波に反映されているという仮説を立て, 様々な解析方法を用いて検証している (図5.2).

周波数パワースペクトル解析

脳波は局所的に小さな活動部分を形成している．脳波から特徴を抽出するには, この微小な周期活動を単位として扱う．例えば, 臨床において使用されている視察判読法では, 平均電位, 平均周波数, ある時間内における特定周波数の出現時間率などを, 波の間隔や波高などを人間が観察した結果に基づいて求める．計算機を用いた場合は, 標本自己相関関数やAIC (Akaike Information Criterion) を適用して, 人間の視察より高精度のセグメンテーションが可能である．

こうして脳波データからあるデータ区間を切り出す場合, その区間よりも大きな周期を持つ周波数成分やオフセットドリフトなどのトレンドを除去する必要がある．脳波データではデータアクイジションの際のオフセットのほか, 脳以外の発生源電位変動の混入であるアーチファクトなどが考えられる．セグメンテーションされた周期的活動部分のスペクトルピークから, 脳波の周波数推移がわかる．例えばAR法を用いて同定した結果を図5.3に示す．(AR法は, 一般的に脳波の様な不規則信号のスペクトルピーク推定にはFFTよりも優れた結果を示す．なお, AR法を解くのに様々なアルゴリズムが存在するが, ここでは内容について特に触れない．)

この脳波スペクトルピークの周波数分布に対して統計モデルを導入し, 電極部位間での脳波パターンを比較する事ができる．まず, 周波数分布状態の時間推移を調べ, 異なるデータ間での比較

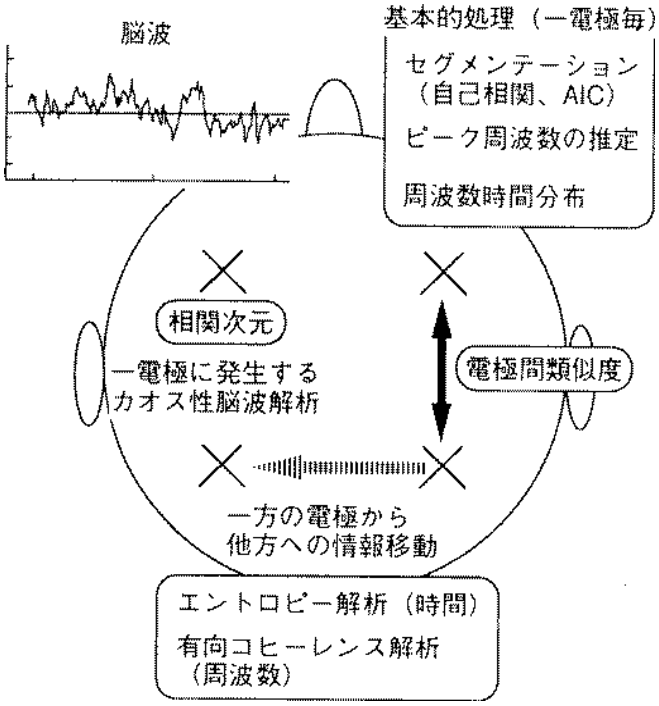


図 5.2: 脳波解析法

を可能にするため適当な大きさの時間ユニットを考える．いろいろな条件における脳波に対して，時間ユニットの大きさを変化させたときの時間ユニット内における分散の平均およびその分散の時間ユニット間における分散の値を調べたところ，10 秒前後から安定する傾向にあった．従って，時間ユニットの大きさを 10 秒とし，音楽の大まかな変化を調べる (図 5.4)．またこの結果を用い，SIM を全部の時間ユニットに対して積算することにより，全体的な周波数パターンの類似度が得られる (図 5.5)．

相関次元の解析

前節で紹介したスペクトル解析の手法は，脳波のような不規則信号が多くの正弦波振動の線形な重ね合わせで構成されることを仮定している．一方，不規則信号が少数自由度の非線形力学系で生成されうるとする見方もある．これは，現在脳に対する最も先鋭的なアプローチの一つである，カオティックで動的な脳情報処理の仮説と結び付く [b20]．脳波が時にカオス性の活動を行なうことは多数報告されている [b29][b30][b31]．最近では，嗅覚が刺激されている際にカオス性活動を行なっている，との研究が発表され注目を集めた [b32]．

カオスは一見複雑かつ確率的であるが，これを分析すると決定論的方程式を反映する構造が明らかになる．この構造を示すのに便利な概念がアトラクタと呼ばれる．アトラクタは連続した $x(t)$ とその時間微分 $(x(t))'$ を用いて描かれる相平面 $(x(t),(x(t))',\cdots)$ の軌跡であるが，微分による高周波雑音の強調を嫌って $(x(t),x(t+\tau),x(t+2\tau),\cdots,x(t+(m-1)\tau))$ (m 次元空間，τ は任意) で表現される．一般的に τ は平均周期の数分の一である．m の値は，相関次元が一定値に落ち着く次元を与える．3 次元で試した例を図 5.6に示す．ストレンジアトラクタの非整数次元を求めるために相関積分を利用する．各条件下における相関次元と埋め込み次元の関係を図 5.7に示す．なお，相関次元の推定には次のような問題点が存在する．

- 相関積分における 2 点間距離は独立ではない．

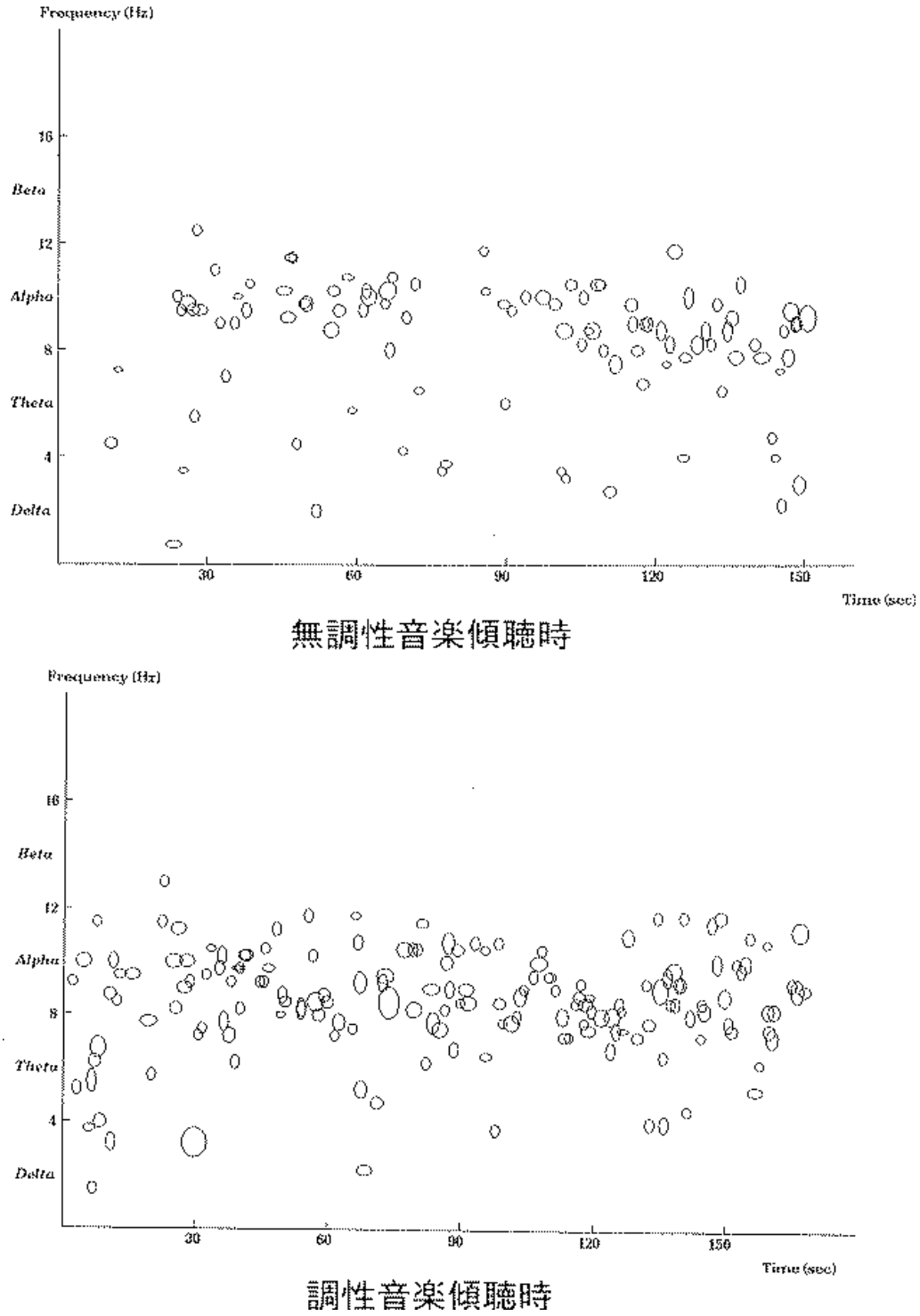


図 5.3: 周波数ピークの比較

- 相関積分は統計的に相関のある情報を含む.
- スケーリング領域の決定と直線のあてはめにおける誤差が存在する.

これに対し, 2 点間距離分布を考慮に入れることでこれらの欠点を補う方法が提案されている [b33]. さらに, 今後は定常性との関係も含め, 時間方向における相関次元の変動について調査する必要がある. また, ピーク周波数の変動や移動が相関次元に及ぼす影響についても, さらに多くのデータから検証する必要がある.

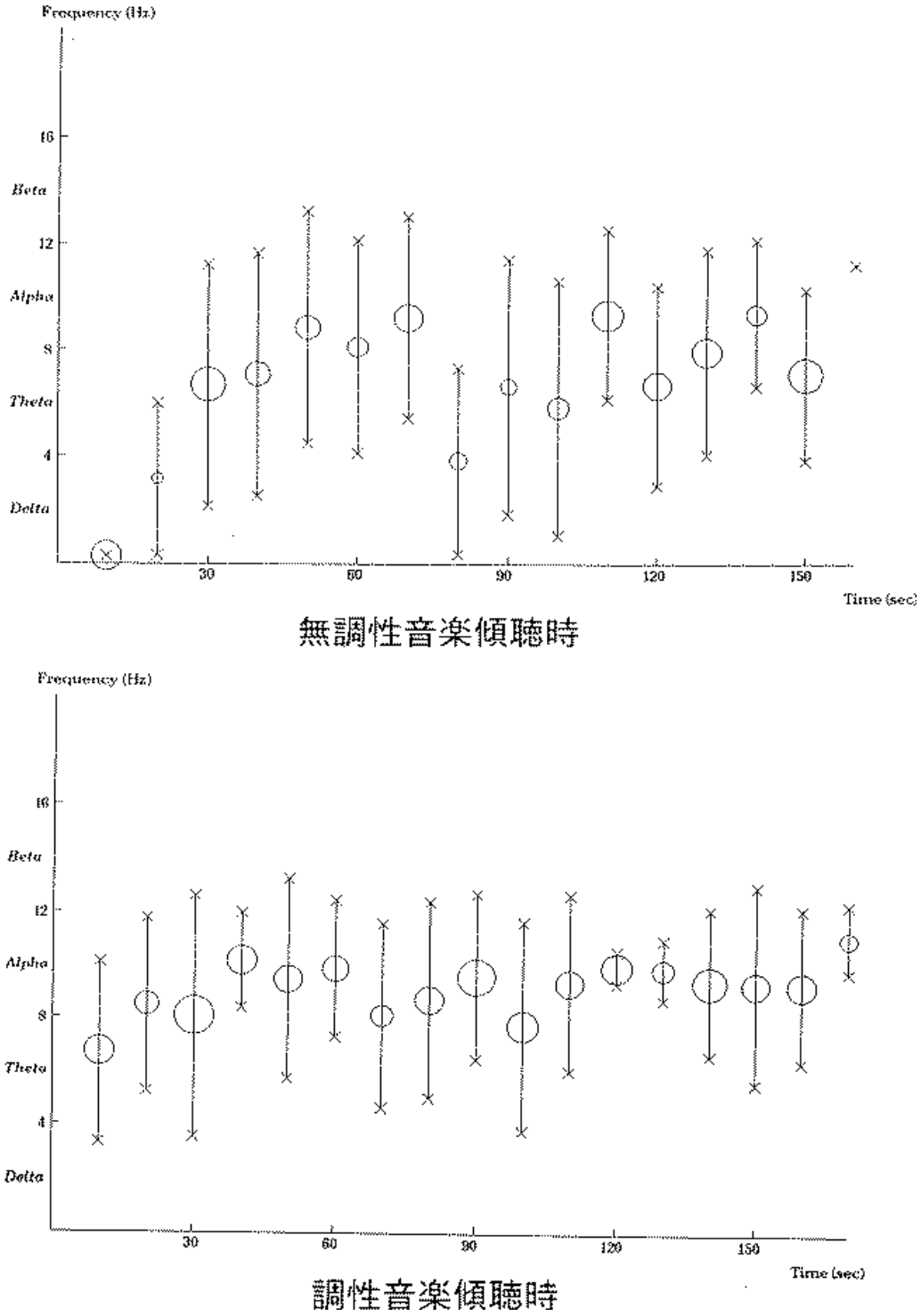
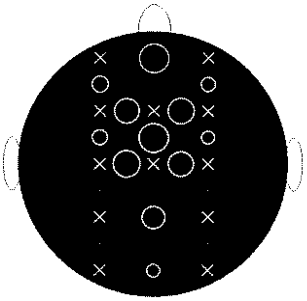


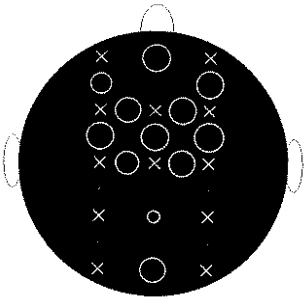
図 5.4: 周波数分散の比較

エントロピー解析

電極間の周波数分布のパターン比較は全体的な類似性の傾向を示しているが, ある特定の情報が移動する様子はわからない. エントロピー解析は, 正規不規則信号がある部位 X から他の部位 Y へ伝播していると考えられるとき方向別に, すなわち X から Y 及び Y から X への相関の度合いを調べる方法である. これによって, 従来使用されているコヒーレンスとは異なって方向が明



無調性音楽傾聴時



調性音楽傾聴時

図 5.5: 共通活動領域の比較

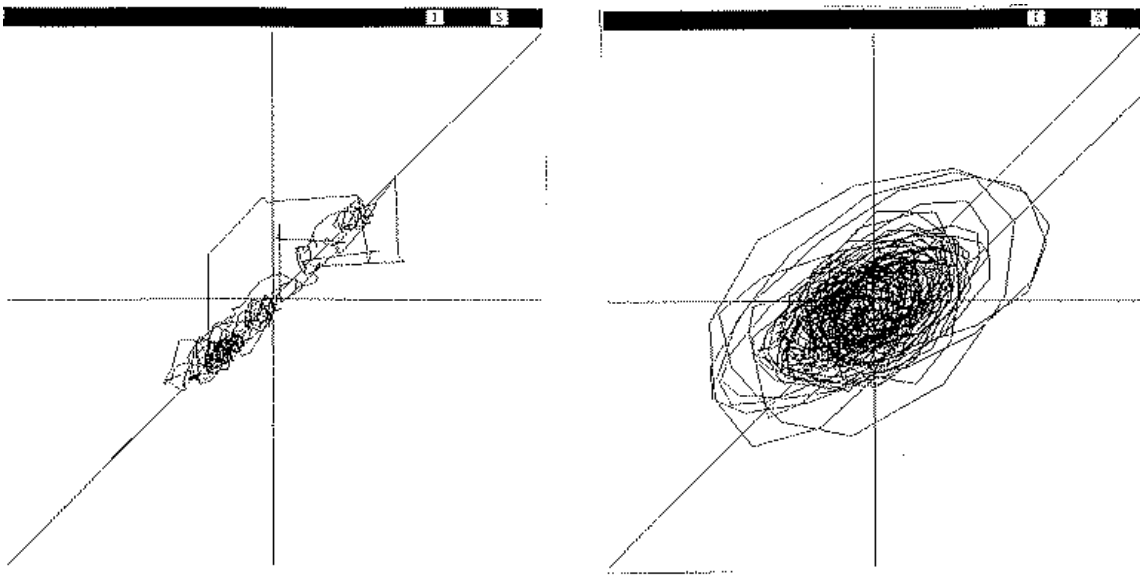


図 5.6: 脳波のアトラクタ

らかになり, またＸとＹの間でフィードバックが存在しても相互相関関数と違って正しい方向がわかる [b34][b35]. 現在, 音楽傾聴時の脳波への利用を検討中である.

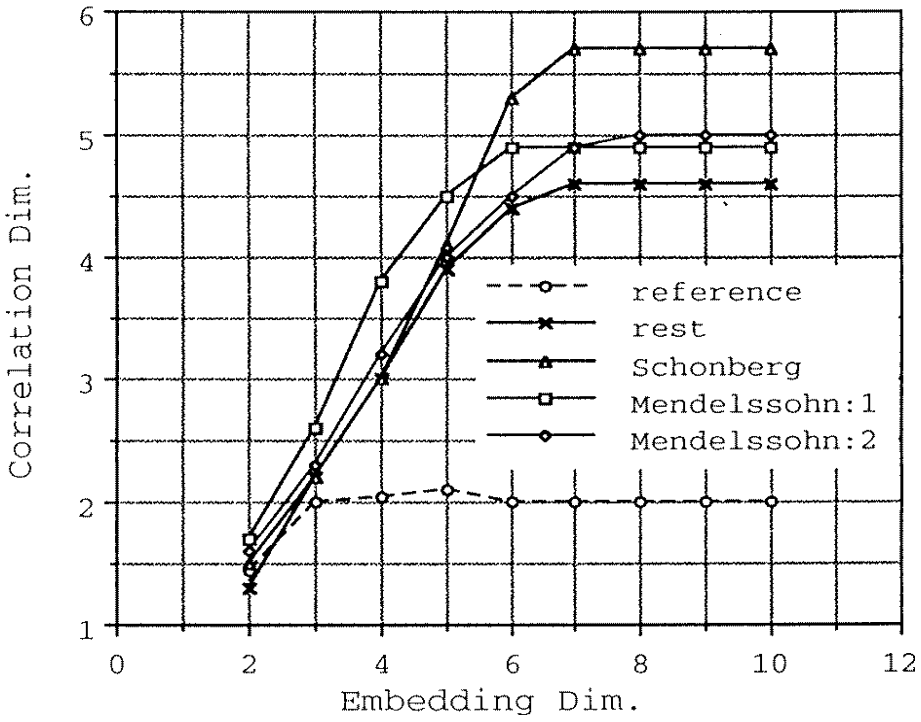


図 5.7: 各条件における脳波の相関次元

例えば, このエントロピー解析法を用いて, 言語刺激時の感覚性言語領野 (Wernicke) と手指運動時の運動領野で大脳半球間において脳波変動の方向を検討した研究がある [b36][b37][b38]. 脳波を 4 つの部位 (感覚性言語領野 (W1) とそれに対応する右半球での部位 (W2), 及び左右後頭部 (O1, O2)) から, 3 つの条件すなわち言語刺激時, 言語として意味のない音声刺激時, 安静閉眼時で記録したデータの解析結果を図 5.8に示す [b36]. 言語刺激時に W1 から W2 への脳波変動の相関度が, 他の部位及び他の条件に比べて大きいことがわかる.

以上, 脳波信号処理について概観した. なお, 時系列解析の手法や AIC のような評価関数については, 文部省統計数理研究所の赤池氏の諸著作に負うところが大である. 「脳と音楽」の研究に直接関係のある内容ではないため参考文献にあげてはいないが, 脳波信号処理の数学的根拠についてご興味のある方は, "赤池弘次, 中川東一郎: ダイナミックシステムの統計的解析と制御, サイエンス社, 1972" 等をご参考にされたい.

5.3.3 PET と MRI を用いた研究例 [b39]

McGill 大学のサージェントらは, PET(陽電子放射断層撮影法) と MRI(核磁気共鳴映像法) を用いて, 楽譜を見る場合や楽譜を見ながら演奏する場合, スケールを聞く場合等様々な条件下における脳活動を分析した. そして, 例えば楽譜を読みながら演奏する場合と単に楽譜を読む場合の結果の差分を取ることにより, 演奏する場合に特有の活性化領域を調べた.

その結果, 楽譜を読んで演奏する過程では言語領域がほとんど活性化されず, 病気や怪我で失語症になったピアニストでも演奏活動が可能な場合もある事実を説明する事ができた. また, 音という同じ媒体を用いていても, 伝達される経路が異なることも明らかになった. なお, 被験者はすべて高度の音楽習熟者であった.

一方, PET や MRI と並んで今後注目される可能性が高い脳磁波を用いた解析に関しては, 今

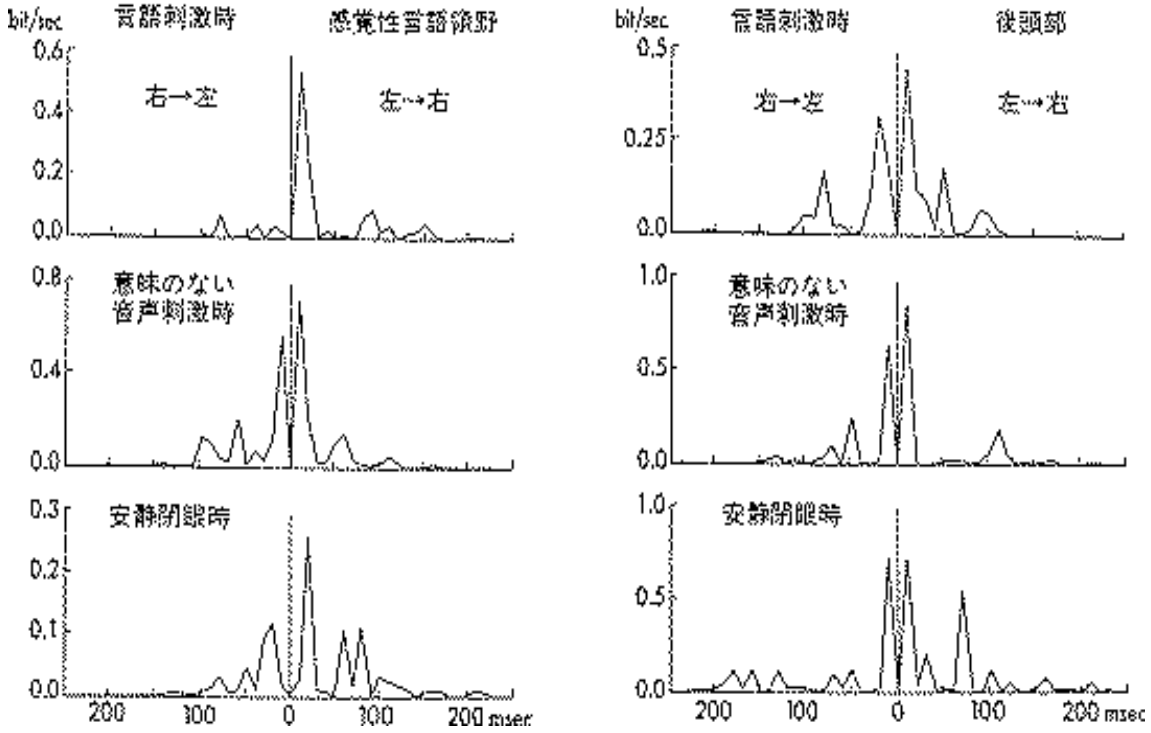


図 5.8: エントロピー解析法の例

のところ音楽を刺激とする研究例がほとんどない．多チャンネルの脳磁波を用いれば，活動源の位置や変化についてよりくわしく調査することが可能となるが，そのためには正しいダイポールの推定法を開発する必要があり，今後の課題とされている．

5.3.4 番外編: ICMC にも登場した脳波 !?

脳波の研究が，主に生理学的立場から行なわれているのはここで紹介した通りであるが，第5.3.1節で紹介したような音楽情報処理の立場からの研究も全くないわけではない．例えば，過去の ICMC (International Computer Music Conference) において，数学者のレム睡眠時 (逆学習の真最中?) における脳波を両半球から記録し，それを可聴周波数帯域に変換して聞かせるパフォーマンスが発表された事がある [b40]．但し，このような研究では脳波そのものの意味よりも，脳波をいかに音に変換するか，という部分に妙味がなければならない．これは，いわゆる音楽情報処理のテーマとしては興味深い．

そういう点に注目すれば，Psychic Lab が開発した IBVA (イーバ) という簡易型脳波記録装置は，脳波を音楽に変換するための手軽なツールとしてパフォーマンスに利用できる可能性を持つ．このシステムが優れているところは，手軽にリアルタイムで脳波を取り込めるだけではなく，FFT による周波数解析も行なった上に，任意の周波数成分の各種パラメータと MIDI 情報をリンクさせて脳波解析結果そのものを音楽に変換できる事である [b41]．また，意識の集中によって音列を制御でき，安静状態では同じシーケンスの繰り返しが生成される事もあるので，得られた MIDI 情報を加工すれば脳波のストレンジ・アトラクタを音で聞くことができる !? もっとも，IBVA から取り込んだ脳波は，研究に必要な厳密性を必ずしも満たしていない事に注意する必要がある．

いずれにせよ，音楽を創るという立場と人間と音楽の関係を解明するという立場では，同じ脳波を用いていても全く視点の異なる研究になる事がおわかり頂けると思う．もちろん，どちらが正しい，という問題ではない．

参考文献

[b1] 加藤, 井口: “小節単位処理に基づいたピアノ楽譜の自動認識”, 電子情報通信学会論文誌, Vol. J71-D No. 5, 1988

[b2] 片寄, 井口: “知的採譜システム”, 人工知能学会誌, Vol. 5 No. 1.

[b3] 和気, 加藤, 才脇, 井口: “演奏者の感情を考慮した協調型演奏システム -JASPER-”, 音楽情報科学研究会夏のシンポジウム'92 予稿集.

[b4] O. シュペングラー: “西洋の没落 - 形態と現実と - ”, 五月書房, 1989.

[b5] 細川周平: “音楽の記号論”, 朝日出版社, 1981.

[b6] Victor Zuckerkandl: “THE SENSE OF MUSIC”, Princeton Univ. Press, 1959.

[b7] Leonard B. Meyer: “Explaining Music”, 1973.

[b8] Fred Lerdahl, Ray Jackendoff: “A Generative Theory of Tonal Music”, 1983.

[b9] 村尾: “楽曲分析における認知”, 音楽と認知, 東京大学出版会, 1987.

[b10] 平賀: “音楽認知のための知識表現”, 音楽と認知, 東京大学出版会, 1987.

[b11] P. N. Johnson-Laird: “THE COMPUTER AND THE MIND”, Harvard Univ. Press, 1988.

[b12] Douglas R. Hofstadter: “METAMAGICAL THEMAS”, 1985.

[b13] 平田: “計算機のための音楽でない音楽”, 情報処理学会誌, Vol. 32, No. 1, pp42-44, 1991.

[b14] M. Nishijima: “Interactive Music Composer Based on Neural Networks”, Proc. of Intl. Computer Music Conference, pp.53-56, 1992.

[b15] R Douglas Riecken: “A K-LINE MODEL OF MUSICAL CREATIVITY: Musical Composition by Emotional Computation”, Proc. of 2nd Intl. Workshop on Artificial Intelligence and Music, IJCAI-89, pp.88-94, 1989.

[b16] F. E. Bloom, A. Lazerson and L. Hofstadter: “BRAIN, MIND, AND BEHAVIOR”, 1985.

[b17] M. Critcheley, R. A. Henson: “Music and the Brain”, 1977.

[b18] 角田: “右脳と左脳”, 小学館, 1992.

[b19] 塚原: “脳の可塑性と記憶”, 紀伊国屋書店, 1987.

[b20] 津田: “カオス的脳観”, サイエンス社, 1990.

[b21] 伊藤, 鳥居, 木谷, 杉下: “特集 - 意識と脳”, 生体の科学, Vol. 43, No. 1, pp4-53, 1992.

[b22] S. Katayama, Y. Hori, R. Nanba, M. Yanagihara and S. Inokuchi: “Correlation between the variance in physiological functions caused by listening to music and divergence in the histories of musical training of University students”, NEUROSCIENCES, 17, pp341-347, 1991.

[b23] M. Osaka: “PEAK ALPHA FREQUENCY SHIFTS DURING MUSIC PERCEPTION”, Proc. of Intl. Conf. on Music Perception and Cognition, pp.267-270, 1989.

[b24] 品川: “クラシック音楽の効用”, 医学講座対談集第 8 集, 日本医師会, 1992.

[b25] T. Oohashi, E. Nishina, N. Kawai, Y. Fuwamoto: “HYPERSONIC EFFECT BY HIGH FREQUENCY SOUND ABOVE THE AUDIBLE RANGE”, Proc. of Int. Symposium on Musical Acoustics, pp119-122, 1992.

[b26] Michel Denis: “LES IMAGES MENTALES”, Presses Universitaires de France, 1979.

[b27] S. Katayama, Y. Hori, R. Nanba, S. Inokuchi: “hanges in the EEG and Circulatory Functions during Musical Tasks”, Neurosciences, Vol. 16 No. 1, pp171-175, 1990.

[b28] Naoki Saiwaki, Seiji Inokuchi, Ken Inoue, Sho Katayama, Yasuo Hori: “An Analysis of Electroencephalographs Recorded During Music Listening”, Proc. of Stockholm Music Acoustics Conference'93, (To be submitted).

[b29] E. Basar: “Chaos in Brain Function”, Springer-Verlag, 1990.

[b30] 品川, 瀬野: “脳波とフラクタル”, 生物物理, Vol. 31, No. 6, 1991.
[b31] Claire D. Koch, Reinhard A. Palovcik, Basim M. Uthman, Jose C. Principe: “Chaotic Activity During Iron-Induced “Epileptiform” Discharge in Rat Hippocampal Slices”, IEEE Transactions on Biomedical engineering, Vol. 39, No. 11, Nov., pp1152-1160, 1992.
[b32] Walter J. Freeman: “The Physiology of Perception”, SCIENTIFIC AMERICAN, February, 1991.
[b33] 池口, 合原, ジュッド, 的崎: “脳波の相関次元”, 電子情報通信学会春季全国大会講演論文集, D-162, 1992.
[b34] T. Kamitake, H. Harashima, H. Miyakawa and Y. Saito: “Entropy analysis of the electroencephalogram”, Technical Group on Medical Bioengineering, IECE, Japan, MBE77-47, Dec., 1977.
[b35] Ramesh K. Kushwaha, William J. Williams, Howard Shevrin: “An Information Flow Technique for Category Related Evoked Potentials”, IEEE Transactions on Biomedical engineering, Vol. 39, No. 2, Feb., pp165-175, 1992.
[b36] Inoue, T., Yagasaki, A., Takahashi, H. and Shinosaki, K.: “The dominant direction of interhemispheric EEG changes in the linguistic process”, Electroenceph. clin., Neurophysiol., 51, pp265-275, 1981.
[b37] Inoue, T., Shinosaki, K., and Yagasaki, A.: “Interhemispheric EEG interactions during voluntary movement”, Neuropsychologia., 20, pp219-224, 1982.
[b38] Inoue, T., Shinosaki, K., and Yagasaki, A.: “The direction of spread of alpha activity over the scalp”, Electroenceph. clin., Neurophysiol., 55, 290-300, 1983.
[b39] J. Sergent, E. Zuck, S. Terriah, B. MacDonald: “Distributed Neural Network Underlying Musical Sight-Reading and Keyboard Performance”, SCIENCE, Vol. 257, July, 1992.
[b40] B. Streitberg, K. Balzer: “THE SOUND OF MATHEMATICS”, Proc. of Intl. Computer Music Conference, pp158-165, 1988.
[b41] “イーバ ユーザーマニュアル”, ランダムエレクトロニクスデザイン, 1990.