音楽情報科学　39－17
（2001. 2. 23）

演奏フィードバックを用いた伴奏システム

福井 浩司　堀内 靖雄　市川 熹

千葉大学 工学部 情報工学科

263-8522 千葉県 千葉市 稲毛区 弥生町 1-33

fukui@icsd4.tj.chiba-u.ac.jp

あらまし　伴奏システムとは人間の独奏者の演奏テンポや音量に追従してコンピュータが伴奏パートを演奏するシステムである。従来の伴奏システムは人間の独奏者の演奏のみを聞いて伴奏システム自身が発音した演奏を聞いていなかった。そのため伴奏システムは自分自身の音の発音時刻や音量を誤認識していた。本研究ではその原因とそれによって引き起こされる問題点について考察し解決策としてフィードバック機構を提案する。フィードバック機構を用いて伴奏システムに自分自身の音をきくための「耳」を装備する。耳を使い正確な発音時刻と音量を認識することで演奏遅延の補償、演奏の正確なずれの認識、音量の追従といったことを実現する。

キーワード

The Accompaniment System with Feedback

Koji Fukui　Yasuo Horiuchi　Akira Ichikawa

Department of Information Sciences,
Faculty of Engineering, Chiba University

1-33, Yayoi-cho, Inage-ku, Chiba-shi, Chiba, 263-8522, Japan

fukui@icsd4.tj.chiba-u.ac.jp

Abstract　Past accompaniment systems could only hear the performance by human soloists and could not hear system's own performance. Therefore the system couldn't know the actual onset time of its own note, the accurate time lag to the soloist and the loudness of its own performance in the real world. This unawareness causes some problems for natural sounding ensemble. In this study, we introduce the feedback mechanism in order to give an ear to the accompaniment system so that the accompaniment system is allowed to compensate the any kinds of delay, recognize the accurate time lag to the human performer and match the soloist in terms of their loudness. The prototype system was developed using the PC with a DSP card.

key words

1 はじめに

伴奏システムとは「あらかじめ楽譜が与えられている状況で、人間の独奏者の演奏に協調してコンピュータが伴奏パートを演奏するシステム」である。

従来のコンピュータによる演奏はテープなどに録音されるものであったため、人間と合奏をおこなうためには人間が機械に追従しなければならなかった。このような演奏形態を機械主導型のアンサンブルと呼ぶ。機械主導型のアンサンブルでは人間はテンポや音量といった面で非常に制限された表現しかできなかった。これは我々に馴染み深いカラオケといったものでも同様である。

しかし伴奏システムを用いた独奏者と伴奏システムが互いに協調しながら演奏するインタラクティブな演奏形態の実現が望まれる。このような演奏形態は機械主導型アンサンブルに対して協調型のアンサンブルということができる。協調型のアンサンブルでは相手の演奏が自分の演奏に影響を与え、それがまた相手の演奏に影響を与え、というように互いに演奏のやりとりをし合う。このやりとりは音楽の楽しみの一つであるだろう。

本研究ではより自然な協調演奏をおこなうためのフィードバック機構を提案する。

2 従来のシステム

2.1 従来のシステムの構成

従来のシステムの構成について文献 [5] の流れを参考にして簡単に述べる。マイクから入力された独奏者の演奏は信号処理により楽音信号に変換される。あるいは MIDI 楽器から直接楽音情報を得る。楽音情報はホストコンピュータに送られ独奏者の演奏の追跡、伴奏パートのスケジューリングがおこなわれる。計画された伴奏パートの演奏の楽音情報は MIDI 音源に送られ、波形合成されてスピーカから出力される（図 1)。

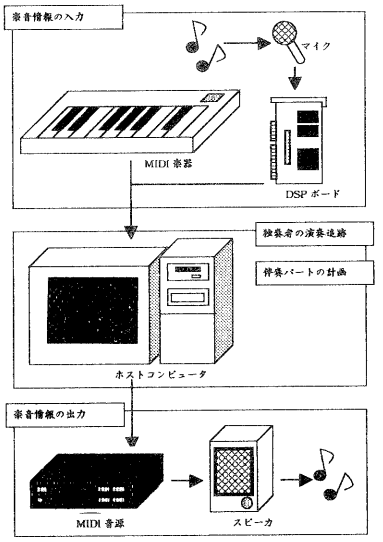

図 1: 使用機器の構成

2.2 問題点

従来の伴奏システムでは独奏者の演奏のみから情報を取得し、伴奏システムの演奏から情報を得ることは想定していない。これは一見当然のことである。なぜならシステムは自らの演奏出力情報を持っているわけでわざわざ発音した伴奏の波形から再取得する情報は何もないように思える。しかし実際にはさまざまな理由によって予定していた楽音信号と発音された楽音信号は異なってしまう。以下、その原因とそれによって引き起こされる問題点について示す。

2.2.1 伴奏システム側の発音時刻の誤認識

伴奏演奏の出力において、MIDI 音源がホストコンピュータから受け取った MIDI 信号を実際に解釈して、対応する波形を合成して出力するまでにはある時間を要する（図 2)。文献 [1]、[2] によると今回実験に使った Roland 社の SC-88VL という MIDI 音源では最低で 5ms、最大で 15ms 程度の遅延が生じる。しかもその遅延時間は音の高さや発音数などによって変動し、一定ではない。

そのため実際の発音時刻はシステムが予定した時刻よりも遅くなっている。それにもかかわらずシステム自身はそれに気づくことができない。これがコンピュータ単体で演奏する場合ではすべての楽音が遅くなるのでそれほど演奏は破綻しない。しかし人間と機械との合奏の場合では互いの相互作用によって演奏が意図したものに対してかけ離れてしまう。

この様子を図 3 を例に説明する。協調演奏では互いが相手の演奏速度に合わせようとして演奏をおこなっている。伴奏システムは人間に合わせようと演奏計画する。そして伴奏システム側からは人間と同時のタイミングで発音できたとする。しかし実際の演奏は遅延してしまうため人間側では発音を早くしてしまったと感じる。人間側も伴奏システムに合わせようと演奏計画するため人間はテンポを落とす。そして次のタイミングでは人間は伴奏システムと同時のタイミングで発音できたとする。しかしそれは遅延した実際の演奏に対して同時ということで伴奏システム側から見れば発音を早くしてしまったと感じる。そして伴奏システムは人間に合わせようとテンポを落とす。このようなやりとりが繰り返され結果として演奏テンポがひっぱられ、遅くなってしまう。あるいは人間の独奏者が演奏が遅くならないよう、むりにリードするため不自然な演奏になってしまう。

もう一つ問題となる点は独奏者と伴奏システムの発音時刻の正確なずれがわかっていないということである。文献 [3],[4] によると人間は互いの演奏のずれなどの情報を利用して自分自身の演奏を修正しているということが示唆されている。そのため伴奏システムにも互いの正確なずれの情報が必要である。しかし MIDI 音源での遅延の問題があり伴奏システムは実世界で生じた正確なずれを認識することはできない。

そのため正確なずれの情報を利用した演奏制御をおこなうことはできない。

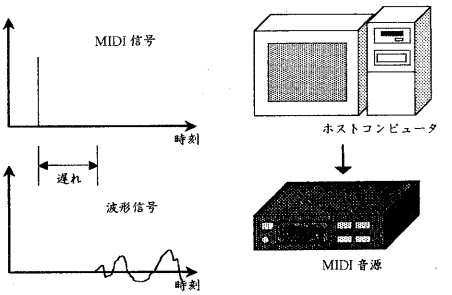

図 2: MIDI 音源の波形合成における遅延

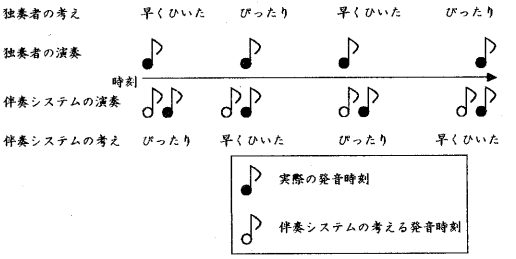

図 3: MIDI 音源の遅延によるテンポの遅れ

2.2.2 音量の誤認識

MIDI 信号における音の強さに相当する Velocity 値[1] と実際に発音される音のパワーの大きさといったことは MIDI 規格で決められているわけではない。使用する MIDI 音源が変われば実際に発音される音のパワーは変わってしまう。もし同じ音源、同じ Velocity 値であったとしても使用する音色などによって知覚される音量は変わる（図 4）。このことは伴奏システム自身が意識している音量と独奏者が実際に聞く音量とが一致していないということである。

このような状態では独奏者と伴奏者が互いに音量を上げ続けたりもしくは逆に下げ続ける、といったことがおこる（図 5）。あるいは人間の独奏者が音量が変わらないように努めるために、独奏が大きすぎたり伴奏が大きすぎる状態で演奏が続き、不自然な演奏となる。

[1] 厳密には鍵盤を押さえる速さに相当するパラメータ

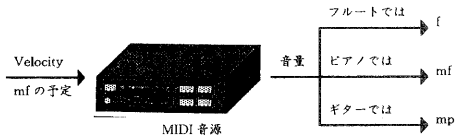

図 4: MIDI 音源の Velocity と音量

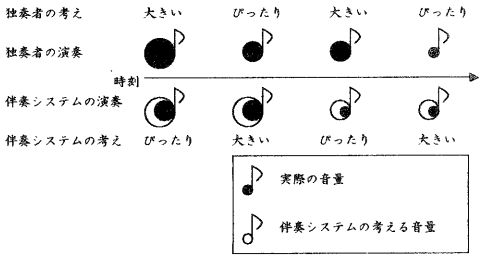

図 5: 音量の誤認識による影響の例

3 フィードバック機構

3.1 フィードバック機構とは

従来のシステムでは伴奏システムが自分自身の音を聴いていないためにさまざまな問題がおきていた。それは当然のことで人間の場合でも自分自身の音が聴こえない状態[2]で伴奏をおこなえばすべてを勘に頼ることになる。そのような状態では独奏者の演奏に追従できているのか不安であるし、自分が演奏を間違ったとしても気づけない。機械の場合も同様で、演奏で音程などを間違うことはないにしても音量、発音時刻などは勘で演奏している状態である。

そこで本研究ではフィードバック機構を用い、伴奏システムに自分自身の音を聴くための耳を装備する。フィードバック機構とは MIDI 音源から出力された波形データを DSP ボードの入力へフィードバックし発音時刻、音量を認識する機構である（図 6）。

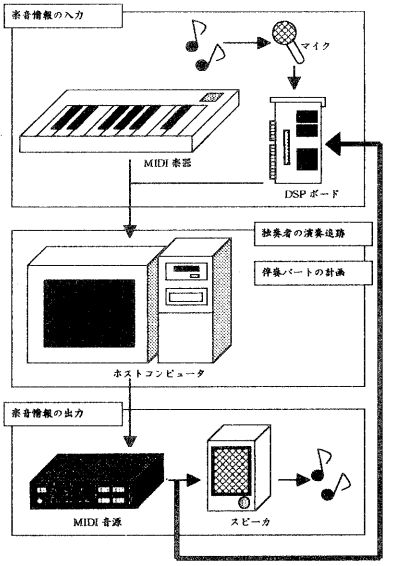

図 6: フィードバック機構の構成

3.2 フィードバック機構を用いたシステムの構成

本研究のプロトタイプシステムでは、ホストコンピュータに IBM 社の Aptiva(Pentium166MHz)、MIDI 音源に Roland 社の SC-88VL、DSP ボードに TexasInstrument 社の TMS320C44 をコア CPU にもつ、MTT 社の HERON DSP6040 ボードに、4 チャンネル（ステレオ 2 チャンネル）アナログ入力を持つ A/D 変換モジュールの HERON ADM16-4 を組み合わせて使用した。今回、波形データのサンプリング周波数は入力の音域や DSP 上での信号処理時間などを考慮して 8kHz とした。量子化ビット数は 16bit でモノラル入力としている。また、MIDI 音源やマイクから発音された波形信号は一度ローパスフィルタでナイキスト周波数の 4kHz 以下にした後 A/D 変換モジュールに入力される。ホストコンピュータの OS は Microsoft 社の Windows95 を用いた。

[2]ただし、伴奏者には独奏者の演奏は聞こえている。

3.3 伴奏の発音検知アルゴリズム

伴奏の発音検知はフィードバック機構において核となる部分である。伴奏の発音検知は伴奏の波形データから楽音の発音の部分を検知してその発音時刻と音量を調べるものである。独奏者側の発音検知と出力する結果は同じであるが独奏はモノフォニック（単音）を対象としているのに対し伴奏は通常ポリフォニック（複音）であるため同じアルゴリズムは使用できない。そのかわりに伴奏では楽音の基本周波数と発音予定時刻の情報が利用できる。

3.3.1 Wavelet 変換

Wavelet 変換では時間的にも周波数的にも相似性を持つような基底によって解析するため短時間フーリエ変換と比べてよい分解能が得られる。また人間の知覚特性に近い性質を持っているため音響信号の解析に向いている [6]。

まず入力された伴奏の波形データを Gabor の基底によく似た Wavelet を使って連続 Wavelet 変換する。通常の Gabor の基底は $\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}e^{ix}$ である。$e^{-\frac{x^2}{2\sigma^2}}$ の部分は時間を決定するための部分で e^{ix} の部分は周波数を決定するための部分となっている。$\frac{1}{\sqrt{2\pi}\sigma}$ は基底を正規化する部分である。今回は計算コストの削減のため周波数を決定する e^{ix} の部分に注目し、時間に関しては方形窓を使う。方形窓の長さは波長のちょうど 6 倍とした。長さを周波数ごとに変化させることで短時間フーリエ変換では得られない時間分解能を実現できる。波長の 6 倍とした理由は十分な時間分解能が得られる範囲でノイズを拾いすぎないように長めに取ったからである。最終的に離散化し、正規化した基底を考えると

$$\frac{f}{8000\times 6}e^{i\frac{2\pi fx}{8000}}$$

となる。ただし、f は調べたい周波数、8000 はサンプリング周波数、x の範囲は $-\frac{8000\times 6}{2f}\sim\frac{8000\times 6}{2f}$ の整数である。

それから位相情報は今回のシステムにとって必要ないため、無視する。振幅の大きさを波形データと上記の基底との内積をとり、そのノルムからもとめる。振幅の対数をとってパワーの情報に変換する。これは人間の知覚特性に近づけるためである。

Wavelet 変換は通常、計算コストが非常に高いが伴奏データの検出では楽音の基本周波数が既知であるので余計な周波数を計算しないことで計算コストを下げることができる。ただし時間精度と信頼性の向上のため、計算する周波数は楽音の基本周波数だけでなく、その 2 倍、3 倍の 3 種類の周波数も考慮する。高い周波数ほど時間の分解能が高くなるので時間精度は上がるが雑音などに影響されやすく信頼性は下がる。特に音楽の性質上、同時に鳴っている楽音の倍音が一致する場合が多く、高周波数の信頼性をさらに下げている。

最終的にこの 3 種類の周波数を解析した結果を総合して発音時刻と音量を決定する。Wavelet 変換は 1ms ごとに行われてパワー情報を次の「発音予定時刻を使った重みづけ」ステージへと送る。図 7 は実際のピアノ伴奏の波形を Wavelet 変換して得られたパワー情報の例である。

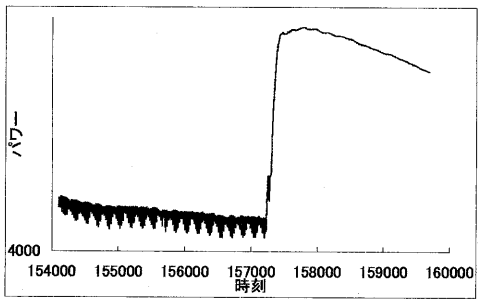

図 7: Wavelet 変換した波形データの例

3.3.2 発音予定時刻を使った重みづけ

発音予定時刻に近いほどパワー変化に敏感に反応し、遠いほど変化を無視する。このようにすることで局所的なパワー変化などを発音として誤認識してしまうことをできるだけ防ぐ。

具体的な方法について。パワー情報を微分し、発音予定時刻でちょうど 1、発音予定時刻から離れるほど 0

に近くなる関数と積をとる。その結果を積分することで発音予定時刻に近いほど変化に敏感な「重みパワー」を得ることができる（図8）。

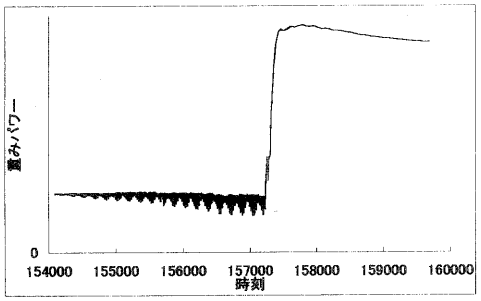

図 8: 発音予定時刻を使った重みづけ

積をとるための関数は今回は

$$1 - \frac{||T-t||}{D}$$

とした。ただし、$||T-t|| < D$ としてそれ以外では0となるようにした。Tは発音予定時刻、tは時刻、Dは探索幅である。Dをリアルタイムに調節することで性能を上げる。つまり、発音予定時刻が信頼できるときにはDを小さくして誤認識を減らすことができ、発音予定時刻が信頼できないときにはDを大きくして発音検知ミスを減らすことができる。今回は直線的な関数を用いたが、今後は関数の形をいろいろと変えることを検討したい。

3.3.3 発音時刻及び音量の算出

音量の算出は重みパワーが最大値をとった時刻での重み無しパワーの値とした。発音開始時刻の算出は重みパワーが最大値をとった時刻とそれ以前の各時刻の重みパワーとを結ぶ直線を引いたときにその傾きが最大となる時刻とした。つまり時刻 t における重みパワーを $W(t)$ とし、重みパワーが最大値をとった時刻を t' とすると

$$\frac{W(t') - W(t)}{t' - t}$$

を最大にする t $(t < t')$ を発音開始時刻とした（図9）。これは人間の発音認識はパワーの立ち上がりの量と立ち上がるまでの時間でおこなっているのではないかという予測から決めた。簡単な発音認知実験をおこなってみたところ確かに立ち上がるまでの時間が短いほど、立ち上がる量が大きいほど発音と感じやすかった。

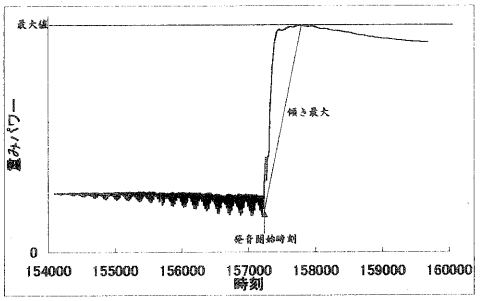

図 9: 発音時刻の推定

このアルゴリズムでは音量をパワーの最大値とした。ピアノなどは比較的そのような傾向がある。しかし一般の楽器では 音量=パワーの最大値 となっている場合は稀である。そのためこのアルゴリズムを様々な楽器に適用することは危険であろう。また発音時刻は音の立ち上がり時間から推定しているためバイオリンなど立ち上がりがなめらかな楽器での発音時刻は当然信頼性が低くなるだろう。以上よりこのアルゴリズムはピアノに特化した発音検知と考えたほうがよい。ただし、通常伴奏システムはピアノ伴奏を想定しているため、それほど大きな問題とはならないであろう。

検知された発音時刻と音量はホストコンピュータに送られて利用される。

4 フィードバック機構の利用

4.1 演奏テンポの遅れの修正

第2.2.1節で述べたように、ホストコンピュータからMIDI音源に制御信号を送ってからMIDI音源が音を発音するまでには十ミリ秒程度の遅延がある。そのため実際の発音は従来のシステムが考えている発音時刻よりも遅れてしまう。その結果演奏テンポが徐々に遅れてしまったり不自然な演奏となるのであった。そ

してこの原因は伴奏システムが自分自身の楽音を聴いていないことから起こっていた。

本システムではフィードバック機構によりシステムの正確な発音時刻の情報を得ている。ホストコンピュータではDSPボードから報告された発音の正確な時刻とその音を発音した予定時刻を用いて発音遅延時間を把握することができる。発音遅延時間の履歴から発音遅延時間を予測する。そして次の楽音からは予測発音遅延時間分だけ早く演奏を計画することで目的の時刻に発音がされるように努力する。今回のシステムでは単純に過去16回分の楽音の遅延時間を平均してそれを予測発音遅延時間とした。但し、発音楽音数や音色の状態、楽音の周波数などによって遅延時間は変わるためそのような情報を活用するとよりよい予測ができるかもしれない。今のところさまざまな状態における遅延時間の傾向がわかっていないことと使用するMIDI音源によって傾向が変わることが予想できるためそのような情報は利用しなかった。

以上のような処理をおこなっても毎回遅延時間は変わるため、完璧に目的の時刻に発音できているわけではない。しかしながら平均の遅延時間を0にすることはできる。平均の遅延時間が0であればテンポを速める量とテンポを遅くする量が釣り合うため総合的にはテンポを変化させることなく演奏が進む。また次節で述べるように、その誤差が生じたとしても誤差を正確に認識できるため、次の演奏を決定する際はその誤差の影響は無視できる。

4.2 演奏のずれの正確な認識

従来はMIDI音源の発音遅延の問題で伴奏システム自身が発音した楽音の正確な発音時刻はわかっていなかった。そのため伴奏システムには独奏者と伴奏者との演奏の正確なずれは認識されていなかった。

本システムではフィードバック機構により伴奏システムの正確な発音時刻の情報と独奏者の発音時刻の情報がある。これらの情報から独奏者と伴奏者の演奏のずれの情報を正確に計算できる。文献[3]、[4]には1拍前の「両者の時間的ずれ」と「両者のテンポのずれ」から人間の未来の演奏を決定するモデルが提案されている。本研究でもこのモデルを採用し伴奏システムの演奏を決定する。1拍前のずれDとテンポのずれTから時間長変化Yを推定するモデルは

$$Y = \alpha D + \beta T$$

と書ける。そしてずれが大きいときにはずれを重視するように係数αを動的に変化させて決定する。βは定数で固定とする。本プロトタイプシステムでは、文献[4]を参考にして、$\alpha = 7||D|| + 0.4$、$\beta = 0.9$とした。

4.3 音量の追従

第2.2.2節で述べたように、MIDI信号におけるVelocity値とMIDI音源から発音される楽音の音量とに決まりがあるわけではない。そのため同じVelocity値でも音源や音色が変われば知覚される音量も変わってしまう。

本システムではフィードバック機構で伴奏システムの音量の情報を得ている。これと独奏者の音量を比較する。ただし音楽の場合、独奏者と完璧に同じ音量であればいいというわけではない。例えば独奏者がメロディを演奏するような場合では伴奏システムのほうが小さい音量となるであろう。あるいは独奏と伴奏とのかけあいなどでは独奏と伴奏が交互に大きな音量となるだろう。つまり音楽の状況によって最適な音量は違う。そこで音量の比率を演奏スケジュール（楽譜ファイル[3]）に記述しておくことにする。システムは比率が保たれるように伴奏計画をおこなう。例えば演奏スケジュールで独奏者の音量を1500、伴奏システムの音量を1000とした場合に実際の独奏者の音量が1200だったとする。そのときは伴奏システムの音量を$\frac{1200\times1000}{1500} = 800$で演奏をおこなう。演奏スケジュールから予測される独奏者の音量と実際に独奏者により演奏された音量との比率は過去4拍の値を平均して、伴奏の演奏に用いる。

次に、同じVelocity値でも音色などによって知覚される音量が変わってしまう問題を考える。本システム

[3]本来演奏スケジュールは楽譜ファイルと演奏履歴から作成されるが本システムには演奏履歴を考慮する機構がまだ搭載されていないため演奏スケジュールと楽譜ファイルは同じものである。

ではVelocity値と知覚される音量には線形関係があると仮定する。そして音量$=a\times Velocity+b$という式で近似する。a、bは過去16音のデータを用いて最小二乗法で推定する。

このような処理を入れることで伴奏システムの音量が大きすぎたり小さすぎる状態のまま演奏が続くことを防ぐことができる。今回はVelocity値と知覚される音量に線形関係を仮定したがこれを改良すれば音量追従の収束をもっと速めることができるだろう。

5 おわりに

本研究では伴奏システムが自分自身の演奏を聴くことのできるフィードバック機構について具体的な発音検知アルゴリズムを提案し実装した。また、フィードバック機構によって得られた正確な発音時刻、音量の情報を用いて演奏遅延の補償、演奏のずれの正確な認識、音量の追従をする方法を提示した。

今後の課題としては本文中でもいくつか述べたがまず発音検知アルゴリズムの改良があげられる。音量の算出にはパワーの最大値を使っているがこれは使用する楽器によるため危険であり音量の追従をおこなう際に問題となるであろう。パワーの積分値を使うといった工夫が必要であると考えられる。それからVelocity値と音量との関係を比例と仮定したため目的の音量に辿り着くまでに数ターンかかっていたがこれをもっとよい式で近似することですばやく目的の音量を得ることができるだろう。発音時刻の推定ではパワーの立ち上がりの情報を利用しているためパワーの立ち上がりがなめらかな楽器への対応が望まれる。

また、今後の展望としてより高度な耳の機能を持たせることが考えられる。例として独奏者のビブラートの幅を計測し伴奏システムのビブラートをその幅に合わせるといったことがあげられる。

そのほかフィードバック機構とは関係がないがよりよい演奏の予測、音量の追従といった意味では各パートに独立度のパラメータを取り入れて独立度の高いパートに追従するようなシステム、あるいはリハーサルを行って演奏の傾向を学習するシステムは非常に効果があるだろう。

参考文献

[1] 堀内靖雄, 演奏フィードバックを利用した伴奏システムの時間制御, レクチャーノート／ソフトウェア学18「インタラクティブシステムとソフトウェアV」(尾内理紀夫編), pp.31～36, 近代科学社,1997

[2] 高橋岳樹 堀内靖雄 市川熹, フィードバックを利用した伴奏システム, インタラクション'99論文集, pp.59～60,1999

[3] 坂本圭司 堀内靖雄 市川熹「計算機との合奏データによる人間の演奏モデルの推定」, 情報処理学会研究報告 Vol.2000 No.94 pp.57-62,2000

[4] 石毛大悟 堀内靖雄 市川熹「相互作用を考慮した人間の協調演奏モデルの推定」, 情報処理学会研究報告 Vol.2000 No.94 pp.63-68,2000

[5] 堀内靖雄,『自動伴奏』, 長嶋洋一 橋本周司 平賀譲 平田圭二 (編),「コンピュータと音楽の世界」,pp.252～269, 共立出版, 1998

[6] 青柳龍也 小坂直敏 平田圭二 堀内靖雄 (訳・監修), コンピュータ音楽ー歴史・テクノロジー・アートー, 東京電機大出版, 2001