(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4360356号
(P4360356)
(45)発行日　平成21年11月11日(2009.11.11)　(24)登録日　平成21年8月21日(2009.8.21)

(51)Int.Cl.　F I
G10H 1/00 (2006.01)　G10H 1/00 102Z

請求項の数 2 (全 21 頁)

(21)出願番号　特願2005-202510(P2005-202510)
(22)出願日　平成17年7月12日(2005.7.12)
(65)公開番号　特開2007-24919(P2007-24919A)
(43)公開日　平成19年2月1日(2007.2.1)
審査請求日　平成20年3月31日(2008.3.31)

(出願人による申告)平成１６年度独立行政法人情報通信研究機構、研究テーマ「超高速知能ネットワーク社会に向けた新しいインタラクション・メディアの研究開発」に関する委託研究、産業活力再生特別措置法第３０条の適用を受ける特許出願

特許権者において、実施許諾の用意がある。

(73)特許権者　393031586
株式会社国際電気通信基礎技術研究所
京都府相楽郡精華町光台二丁目２番地２
(74)代理人　100090181
弁理士　山田　義人
(72)発明者　西本　一志
京都府相楽郡精華町光台二丁目２番地２
株式会社国際電気通信基礎技術研究所内
(72)発明者　大島　千佳
京都府相楽郡精華町光台二丁目２番地２
株式会社国際電気通信基礎技術研究所内

審査官　日下　善之

最終頁に続く

(54)【発明の名称】合奏支援システム

(57)【特許請求の範囲】
【請求項１】
演奏位置と、各演奏位置毎の第１奏者が演奏すべき音の音高データと、各演奏位置毎の第２奏者が演奏すべき音の音高データとを含む楽譜データを格納する楽譜データベースを備え、前記第１奏者の演奏に応じて第１入力インタフェースから出力される第１演奏データに従って発音し、かつ第２奏者の演奏に応じて第２入力インタフェースから出力される第２演奏データのうち音高データだけを前記楽譜データベースに格納された楽譜データに含まれる前記第２奏者が演奏すべき音の音高データに差し替えて発音する合奏支援システムであって、
前記楽譜データベースに格納された前記楽譜データを読み込んで前記第１奏者が演奏すべき音の最低音を検索する検索手段、
前記最低音よりも低い範囲を前記第２インタフェースにおける前記第２奏者の使用可能範囲として決定する決定手段、および
前記決定手段が決定した使用可能範囲を表示する領域表示手段を備える、合奏支援システム。
【請求項２】
前記第１奏者が使用する前記第１入力インタフェースと前記第２奏者が使用する前記第２入力インタフェースは同じ入力インタフェースである、請求項１記載の合奏支援システム。
【発明の詳細な説明】

【技術分野】
【０００１】
この発明は合奏システムに関し、特にたとえば、初心者同士でも簡単に合奏や連弾を楽しむことができる、新規な合奏支援システムに関する。
【背景技術】
【０００２】
子供がピアノのレッスンを受けている場合など、子供と一緒に合奏を楽しみたいという親の声は多い。
【０００３】
このため、最近では子供向けのみならず、大人のための音楽教室や音楽ソフトも流行っている。中でも、なるべく短い時間で各々の大人が弾きたい曲を演奏できるようにするレッスンやソフトが多い。これらは、子供のように月日を重ねて基礎から学ぶことよりも、いち早く裡にある音楽を表現することを優先していると思われる。しかし、子供といろんな曲で合奏できるようになるには、やはり楽譜上の音符通りに楽器を奏でるという基礎から習得しなければならず、長い年月を要してしまう。
【０００４】
また、ピアノのレッスンでは、合奏の１つである連弾を取り入れることが奨励されている。たとえば、ピアノを習い始めて１週目の子供が、「ド」を数個並べただけの楽曲を練習する際にも、先生が連弾のパートナーとして多様な和音を響かせながら付き添うレッスンが行われている。連弾を通じて生徒の練習意欲が高まったという報告や、子供たちがパートナーの演奏を聴くことで自ら「やりたい表現」を自然に演奏に表すようになったという報告もある。どんなに初心者の子供も、連弾の中で先生と「呼吸」を通わせるという、音楽性を育てる上で不可欠な営みを行っている。
【０００５】
しかし、家庭においてレッスンに通い始めの子供と、楽器演奏経験のほとんどない親とのペアで、練習や楽しみとして連弾を行うことは非常に困難である。これは、市販されているほとんどのピアノ連弾曲集が、少なくともペアの一方が中級以上に想定されていることからも伺える。最近ではピアノ学習用として、予めパートナーの演奏データが用意されたマイナス・ワンを多くみかけるようになった。しかし、マイナス・ワンを利用してもこの方法では一人ひとりの演奏行為と実在的な関わりを持たないため、パートナーと「呼吸」を通わせることができない。
【０００６】
一方、近年、人間の演奏者と合奏するシステムの研究が進んでいる。予め楽譜が与えられている音楽を対象とした自動伴奏システムが非特許文献１や非特許文献２に開示される。非特許文献１のものは、演奏者が出力した音高情報だけを用いたアルゴリズムを示し、非特許文献２のものは、音高と一定時間ごとの演奏時刻とを認識して伴奏を変化させる手法を提案している。
【０００７】
さらに、非特許文献３では、より自然な伴奏システムの演奏を目指して、人間とコンピュータの合奏を分析し、コンピュータと人間との「ずれ」から次の「時間長変化」を予測するモデルを提案している。
【０００８】
また、非特許文献４では、人間とコンピュータの相互作用を考慮したモデルの提案が行われている。
【非特許文献１】Dannenberg、 R. : An On-Line Algorithm for Real-Time Accompaniment、Proc. ICMC 1984、 pp. 193-198、 1984
【非特許文献２】Vercoe、 B. : The Synthetic Performer in the Context of LivePerformance、 Proc. ICMC 1984、 pp. 200、 1984
【非特許文献３】堀内靖雄、坂本圭司、市川熹：合奏における人間の発音時刻制御モデルの推定、情報処理学会論文誌、Vol. 43、Ｎｏ．2、pp. 260-267、 2002

【非特許文献４】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、電子情報通信学会春季全国大会講演論文集、 pp. 1. 389-390、 1992

【発明の開示】

【発明が解決しようとする課題】

【０００９】

しかしながら、以上の研究は、すべて、人間とシステム(コンピュータ)とによる合奏を目的とするもので、この発明が向けられる、人間同士特に初級者同士が合奏するのを支援するシステムを対象とするものではない。

【００１０】

また、初級者同士の合奏や連弾を容易にするシステムが望まれる。

【００１１】

それゆえに、この発明の主たる目的は、新規な、合奏支援システムを提供することである。

【００１２】

この発明の他の目的は、初級者同士の合奏や連弾が容易に行える、合奏支援システムを提供することである。

【課題を解決するための手段】

【００１３】

請求項１の発明は、演奏位置と、各演奏位置毎の第１奏者が演奏すべき音の音高データと、各演奏位置毎の第２奏者が演奏すべき音の音高データとを含む楽譜データを格納する楽譜データベースを備え、前記第１奏者の演奏に応じて第１入力インタフェースから出力される第１演奏データに従って発音し、かつ第２奏者の演奏に応じて第２入力インタフェースから出力される第２演奏データのうち音高データだけを前記楽譜データベースに格納された楽譜データに含まれる前記第２奏者が演奏すべき音の音高データに差し替えて発音する合奏支援システムであって、楽譜データベースに格納された楽譜データを読み込んで第１奏者が演奏すべき音の最低音を検索する検索手段、最低音よりも低い範囲を第２インタフェースにおける第２奏者の使用可能範囲として決定する決定手段、および決定手段が決定した使用可能範囲を表示する領域表示手段を備える、合奏支援システムである。

【００１４】

請求項１の発明では、第１奏者（プリモ）によって第１入力インタフェース（実施例において相当する部分の参照符号１２´。以下同様。）が操作され、その演奏に応じて第１演奏データとしてＭＩＤＩデータを出力する。同じかまたは別の第２入力インタフェースが第２奏者（セコンド）によって操作され、演奏に応じて第２演奏データとしてＭＩＤＩデータを出力する。第１奏者の演奏による入力インタフェースからのＭＩＤＩデータはそのまま発音手段（１６，１８）から発音される。しかしながら、第２奏者の演奏による入力インタフェースからのＭＩＤＩデータについては、音高データだけが楽譜データに規定されている音高データと差し替えられて発音される。したがって、第１奏者の演奏はそのまま楽音として出力されるが、第２奏者の演奏は音高が差し替えられて楽音として出力される。したがって、第２奏者の実際の演奏の如何に拘らず、第２奏者の演奏の音高は正しく矯正されるので、たとえば、第１奏者が初級者であってかつ第２奏者が初心者であっても、合奏や連弾が簡単に行える。

【００１５】

このように，プリモ側の演奏に対しては一切加工を加えず、そのまま楽音として出力するが，セコンド側の演奏に対しては音高の差し替え処理という加工を加えて楽音として出力するため、基本的にはセコンドはセコンド用の第２入力インタフェースの中のどの鍵盤を押していっても、正しい音高を出力することができる。しかし、たとえば、遠隔で映像をお互いに送りあって行う連弾やレッスンで、このシステムを使用する場合には、セコンドがプリモが使用する範囲の鍵盤を使用してしまうと、同じ鍵盤を重複して使用している映像が流れて混乱を招いたり、対話の中で混乱を招いたりするため、セコンドは使用可能

範囲の中で演奏する必要がある。そこで，請求項１の発明では，検索手段，決定手段，および領域表示手段を設けている。この検索手段（図１１）は、楽譜データを読み出して、第１奏者が演奏すべき音の最低音を検索する。そして、決定手段はその最低音よりも低い範囲を第２入力インタフェースにおける第２奏者の使用可能範囲として決定し、領域表示手段がこの使用可能範囲を表示する。たとえば発光素子（２６）を含む領域表示手段（２４）が第２奏者の使用可能範囲を目視可能に表示する。

【００１６】

したがって、検索手段によって最低音を検索して、領域表示手段によって表示することによって、プリモが使用せざるを得ない範囲とセコンドの使用可能範囲とを事前に把握しておくことができ、第１奏者が最低音より低い位置のキーにタッチするのを防止でき、さらには、入力インタフェースのたとえば鍵盤の使用範囲の重複の問題が生じるのを防ぐことができる。

【００１７】

請求項２の発明は、第１奏者が使用する第１入力インタフェースと第２奏者が使用する第２入力インタフェースは同じ入力インタフェースである、請求項１記載の合奏支援システムである。

【００１８】

請求項２の発明では、入力インタフェース（図１０の１２）の上位領域（図１０の１２ａ）が第１奏者（プリモ）によって操作され、その演奏に応じて第1演奏データとしてＭＩＤＩデータを出力する。同じ入力インタフェース（１２）の下位領域（１２ｂ）が第２奏者（セコンド）によって操作され、演奏に応じて第２演奏データとしてＭＩＤＩデータを出力する。

【００１９】

この請求項２の発明でも、第１奏者の演奏による入力インタフェースからのＭＩＤＩデータはそのまま発音手段（１６、１８）から発音され、第２奏者の演奏による入力インタフェースからのＭＩＤＩデータについては、音高データだけが楽譜データに規定されている音高データと差し替えられて発音手段に与えられる。つまり、セコンド側の演奏に対しては音高の差し替え処理という加工を加えて楽音として出力するので、セコンドは基本的にどの鍵盤を押して演奏しても構わない。しかしながら、プリモとセコンドが同じインタフェースの場合には、セコンドがプリモの演奏範囲の鍵盤を使用するとプリモの演奏の邪魔になる。よって、セコンドが使用できる鍵盤の範囲をセコンドに提示する必要がある。

【００２０】

請求項２の発明では、１台の入力インタフェースを第１奏者および第２奏者が共用する「連弾」ができる。

【発明の効果】

【００２１】

この発明によれば、第２奏者の演奏データの音高だけを差し替えて合奏や連弾が簡単に行えるような合奏支援システムにおいて、検索手段によって第１奏者の最低音を検索して、領域表示手段によって第２奏者の使用可能範囲を表示するので、第１奏者が使用せざるを得ない範囲と第２奏者の使用可能範囲とを把握しておくことができ、たとえば、入力インタフェースでの鍵盤の使用可能範囲の重複の問題が生じるのを防ぐことができる。

【００２２】

この発明の上述の目的、その他の目的、特徴および利点は、図面を参照して行う以下の実施例の詳細な説明から一層明らかとなろう。

【発明を実施するための最良の形態】

【００２３】

図１はこの発明の背景となる合奏支援システム１０を示し、この図１に示す合奏支援システム１０は、プリモ用入力インタフェース１２およびセコンド用入カインタフエース１４を含む。ただし、この図１では２つのインタフェース１２および１４を別々に図示しているが、連弾の場合には、１つのインタフェース（鍵盤）を２つの領域に分けて使用する

ことになる。
【００２４】
なお、「プリモ」とは、ピアノ連弾の場合に高音部を担当する演奏者のことであり、「セコンド」は低音部を演奏する演奏者のことを言う。合奏の場合には必ずしもこのような呼称を用いないが、ここでは、便宜上、合奏をリードする演奏者を「プリモ」と呼び、そうでない演奏者を「セコンド」と呼ぶ。ただし、前者を「第１奏者」、後者を「第２奏者」と呼ぶこともある。
【００２５】
プリモ用入力インタフェース１２およびセコンド用入力インタフェース１４はともに、演奏に応じて、ＭＩＤＩ(Musical Instrument Digital Interface)データを出力する。そして、プリモが演奏したことに応じて入力インタフェース１２から出力されるＭＩＤＩデータは、そのまま、なんら処理を施されずにＭＩＤＩ音源１６に入力され、スピーカ１８から音として出力される。プリモ用入力インタフェース１２からの演奏データ（ＭＩＤＩ）はまた、コンピュータ２０にも同時に入力される。同じように、セコンドが演奏したことに応じて入力インタフェース１４から出力されるＭＩＤＩデータもコンピュータ２０に入力される。
【００２６】
簡単に言うと、コンピュータ２０は、プリモの演奏データと楽譜データとを参照しながら、セコンドの演奏だけを支援する。プリモに対しては演奏支援を行わない。
【００２７】
コンピュータ２０が参照すべき楽譜データは、楽譜データベース２２に記憶されている。
【００２８】
図２がこのような楽譜データの基になる記述形式の一例を示す。この例で、「＊」で始まる行には演奏データそのものが記述されているのではなく、その曲を初心者が演奏する場合に戻りやすい位置を示すマーカを記述するようにしている。
【００２９】
Bloch & Dannenberg（Bloch, J. and Dannenberg, R.B.: Real-Time Computer Accompaniment of Keyboard Performances, Proc. ICMC 1985, pp.279-289, 1985）によれば、演奏の誤りは、以下の３つに分類される。
（1）挿入(extra):演奏された音が楽譜上の音と一致しない(余分な音数)。
（2）音高誤り(wrong):楽譜上の音数通りに音を出しているが、音高が不正確である。
（3）脱落(missing):演奏された音が楽譜上の音の数よりも少ない。
【００３０】
しかし、特に初級者の場合はこれらに「弾き直し」が追加されると考えられる。弾き直しとは、「挿入」と同様に楽譜上の音数よりも多く音が演奏されたケースではあるが、楽譜上に記載されていない音や音高列ではなく、楽譜上に記載されている音や音高列が重複して演奏された場合のことを指す。中級者以上になると、練習中に意識して弾き直しする以外は、基本的に失敗しても演奏の流れを止めずに最後までいこうとする傾向がある。一方、初級者は途中で失敗した場合に戻ろうとする傾向が多い。
【００３１】
ピアノレッスンに通いはじめて４年半の小学４年生である、被験者Ａと、ピアノレッスンに通いはじめて１年の保育園生である、被験者Ｂとによって、いずれもプリモとして、様々な練習曲を初見で、演奏してもらった実験によれば、表１に示す結果が得られた。ただし、実験ではセコンドは、発明者の１人が担当した。表１は、弾き直した回数と弾き直すために戻った個所の特徴である。
【００３２】

【表１】

戻った個所	A	B	合計	%
１拍目	３１５	１４１	４５６	８１．７
第２の強拍	１７	２	１９	３．４
最終拍	０	６	６	１．１
フレーズ開始	５	０	５	０．９
手の替え	４	０	４	０．７
１拍目（１小節前）	２６	２５	５１	９．１
最終拍（１小節前）	２	３	５	０．９
曲の最初	７	３	１０	１．８
段の最初	１	１	２	０．４
合計	３７７	１８１	５５８	１００．０

【００３３】

「１拍目」とは、小節の１拍目に戻ったことを示す。「第２の強拍」とは、４拍子の３拍目や２拍子の２拍目を指す。「最終拍」とは３拍子の３拍目や４拍子の４拍目を指す。以上の３項目は、弾き直しする直前に弾いた音と同じ小節内の拍である。「フレーズ開始」とは、スラーのはじめや、弱起のメロディのはじまりを指す。「手の替え」とは、１つのメロディの中で、弾く手が右手から左手に替った個所に戻ったことを意味する。以上の２項目は「１拍目」「第２の強拍」「最終拍」のカテゴリには数えていない。「段」とは譜面上で横一列に印刷された数小節の塊を指す。なお「１小節前」とは、弾き直す直前の音の１小節前に戻ったことを意味している。

【００３４】

弾き直しの数は被験者Ａは３７７個所あり、被験者Ｂは１８１個所あった。また、戻り位置は小節の１拍目が８割を占めている。

【００３５】

一方、「挿入」は、被験者Ａに１音、被験者Ｂに６音見られた。「音高誤り」は被験者Ｂに37音(延べ１１１音)見られた。「脱落」は被験者Ａに３音見られた。

【００３６】

その他、被験者Ａは小節ごとに楽譜を見て弾く傾向があったため、小節の１拍目の直前に楽譜上には記述されていない空白の時間が起りやすかった。そのため、連弾曲ではセコンドが先走って１拍目の音を鳴らし、再度、被験者Ａと一緒に鳴らすという場面が多くみられた。なお、課題にした独奏曲と連弾曲のレベルや曲数等が違うため、一概には言えないが、被験者Ａは連弾曲のほうが弾き直しが少ない傾向にあった。

【００３７】

この実施例では、上記のような実験の結果を参考にして、図２の「＊」が付された行のように、曲を初心者が演奏する場合に戻りやすい位置をマーカとして記述するようにした。そして、マーカとしては、発明者等の実験では、次の４種類を設定できるようにしている。ただし、１箇所に複数種のマーカを同時に設定することもできる。

【００３８】

Ｂ：小節の頭を示す

Ｐ：フレーズの冒頭を示す

Ｒ：印刷された楽譜の「段」の冒頭を示す

Ｓ：その他、任意に設定可能なマーカ

なお、図１の実施例では、これらのマーカの種類に依存した動作の切り替えは行っておらず、マーカの有無だけをチェックするようにしている。つまり、この実施例では、複数

種類のマーカを用意しているのは、人が楽譜データを作成するとき、あるいは読むときの便宜のためだけであるが，表１に示す弾き直し箇所の頻度を考慮して、これらの種類に応じて重みづけを調整することにより、より精度の高い弾き直し判定を実現することも可能である。

【００３９】

さらに、図２の記述形式では、楽譜データは、次のように記述される。

(1)「＊」で始まるマーカ行以外は、すべて音符データを記述する。１行には、楽譜上で「同時に」発音されることになっている音を列挙する。

(2)「音名」には、音符の「音価」（つまり４分音符、８分音符などの音の長さ）を無視して、音高のみを記述する。たとえば「Ｃ５」、「Ａ３」など。

(3)プリモあるいはセコンドの演奏者が１人で複数の音を同時に演奏する（和音を弾く）場合には、それらの音名を「，（カンマ）」で繋いで列挙する。

(4)プリモとセコンドとの間は「:（コロン）」で繋ぐ。

(5)ある瞬間、一方のみが発音し、他方は発音する音がない場合、発音する音がない方は音高名を指定しない。

(6)「コメント」は記述してもしなくてもよい。

【００４０】

具体例として図３に示す楽譜は、童謡「たき火」の冒頭４小節であり、この楽譜を図２の記述形式でデータ化したものが図４に示される。ただし、図３では、上段がプリモ、下段がセコンドの楽譜である。

【００４１】

図４において、「＊Ｓ」の箇所は、小節の冒頭でもフレーズの頭でもないが、特に戻り易い箇所として設定した例である。楽譜データは上記のように通常のテキストエディタで容易に編集可能な形式であるため、個々の演奏者に応じて、特に戻り易い箇所がある場合は、このように「＊Ｓ」マーカなどを利用者が自由に埋め込んで、自分用に最適化することも可能である。

【００４２】

この図４の記述データに従って、図５に示す楽譜データが、図１の楽譜データベース２２に格納される。

【００４３】

なお、図５の楽譜データで、ポジション（位置）とは楽譜の最初からのプリモとセコンド両方を通しての音符番号を示す。同時に複数の音が演奏される箇所は、それらの音すべてに同じポジション番号を付与する。たとえばポジション１や５などは、それぞれ３つの音に同じポジション番号を割り当てている。プリモまたはセコンドの一方に演奏すべき音があるが、もう一方には対応する箇所に演奏すべき音がない箇所は空欄となる。たとえばセコンドのポジション２、４など。リターンの欄には、楽譜データでいずれかの種類のマーカが設定されたポジションの箇所にのみ「＊」記号がセットされ、他は空欄となる。現実には、Ｃ言語の構造体のリンクによってこのような楽譜データ表を構成している。

【００４４】

このような図１の合奏支援システム１０において、コンピュータ２０は、図６のステップＳ１において、図７に詳細に示す演奏位置判定処理を実行する。この演奏位置判定処理では、簡単にいうと、楽譜データベース２２から取得したプリモ用楽譜データと、逐次入力されてくるプリモの演奏データ（ＭＩＤＩ）とを照合し、現在楽譜上のどの位置（ポジション）が演奏されているかを判定し、この結果を、続くステップＳ３すなわち図８に詳細に示す演奏音高取得処理に通知する。

【００４５】

ステップＳ３すなわち演奏音高取得処理では、簡単にいうと、楽譜データベースから取得したセコンド用楽譜データを参照し、現在のプリモの演奏位置に対応する箇所をセコンドの楽譜上で見いだし、そこからセコンドが今演奏すべき音の音高データを取得し、これをステップＳ５すなわち図８に詳細に示す音高データ差し替え処理に渡す。

【００４６】
音高データ差し替え処理では、簡単にいうと、セコンド用の入力インタフエースから入力された演奏音のＭＩＤＩデータのうち、音高データを指定する値（ＭＩＤＩノートナンバ）のみを、演奏音高取得処理によって取得した音高データに差し替える。この際、その他のデータ(発音時刻や、音の強さに対応するベロシティ値など)はすべてセコンドが入力した値を保持する。
【００４７】
こうして、音高データだけが差し替えられたセコンドの演奏データ（第２演奏データ）がプリモの演奏データ（第１演奏データ）とともにＭＩＤＩ音源１６（図１）に入力され、スピーカ１８から音として出力される。つまり、ＭＩＤＩ音源１６およびスピーカ１８が発音手段として機能する。
【００４８】
したがって、セコンドは鍵盤上のどの鍵を打鍵しても、各時点で演奏すべき正しい音高の音が出力される。一方、音の強弱や発音タイミングなどの音楽表情に関わる要素はすべてセコンドが演奏したまま出力される。この結果、セコンドは楽譜どおりのメロディを容易に再現でき、かつ表情づけした通りに演奏に反映できる。
【００４９】
また、プリモが演奏しているにもかかわらず、セコンドがしばらく休止した場合(打鍵しなかった場合)でも、コンピュータ２０はプリモの演奏データから演奏位置を認識しているため、いつセコンドが再開しても、正確な音高でプリモと合わせることができる。逆に、プリモが演奏を休止したにも拘らずセコンドが打鍵した場合は、１回目の打鍵はプリモが次に演奏する予定の個所に相当した音高が鳴るが、それ以後はプリモが演奏し始めるまで、音が出なくなるように設定されている。
【００５０】
ここで、図７に示す演奏位置判定処理について、詳細に説明する。ただし、この図７は、プリモの打鍵（入力インタフェース１２の操作）というイベントで駆動されるもので、演奏時点ｔｊにおいてプリモがＰｊの音を演奏したものとして表されていることを、予め留意されたい。
【００５１】
演奏位置検出（いわゆるScore Following/Tracking）に関する研究は８０年代より非常に多数なされている。過去のシステムは、「挿入」、「音高誤り」、「脱落」の３種類の誤りに対して対処しているが、いずれも基本的に大きな誤りは起こらない演奏を前提としている。
【００５２】
これに対して、この発明の合奏支援システム１０では、初心者による練習段階でのきわめて不完全な演奏を取り扱う。この場合、上記のように、弾き直しが非常に多く発生し、演奏位置が何度も繰り返して大きく遡ることが頻発する。したがって、従来のシステムではこのような弾き直しに対処できない。
【００５３】
そこで、この実施例では、先の非特許文献１でDannenbergが提案したＤＰマッチングによる手法を拡張し、弾き直しに対処可能な演奏位置検出手法を改良した。
【００５４】
この実施例の演奏位置検出方法では、Dannenbergの手法同様、各音の楽譜上での音価と演奏音の音長は無視して、音高のみのマッチングで演奏位置を判断する。これは、初心者の演奏では演奏時の音長の変動が極めて大きく、楽譜には無い長い停止も頻発するため、音価と音長はマッチングの対象として扱えないという判断に基づく。
【００５５】
プリモ用の楽譜に含まれる音の数をＮとする。ただし、和音のように複数の音が同時に発音される場合、その個所の音数は、同時に発音する音の数に拘らず「１」とし、最高音のみをマッチングの対象とする。さて、プリモが演奏開始からｊ番目の音Ｐｊを演奏した

とき(この時刻を「演奏時点ｔｊ」とする)、前回のマッチングで演奏時点ｔｊ-１での楽譜上の演奏位置がＳｉ（１≦ｉ≦N)の音であると判定されていたとする。このとき、Ｐｊが楽譜上のどの音にあたるかの判定は、以下のアルゴリズムによって行う。

【００５６】

（1）楽譜上のすべての音Ｓｋ(１≦ｋ≦N)の音高Pitch（Ｓｋ）と、Ｐｊの音高Pitch（Ｐｊ）とを比較し、すべての音についてプリモの演奏時点ｔｊにおける重みW（Ｓｋ，ｔｊ）を以下の数１に示す方法で評価する。

【００５７】

【数１】

(a) if Pitch (S_k) =Pitch (P_j)

then W (S_k, t_j):=W (S_{k-1}, t_{j-1}) +1

(b) else W (S_k, t_j):=W (S_{k-1}, t_{j-1}) −1

ただし、W (S_{k-1}, t_{j-1}) −1＜0ならばW (S_k, t_j):=0とする。

【００５８】

このことを図7で示すと、ステップＳ１０１で音数ｋを「１」に設定する。最初はｋ＜N（総音数）であるから、ステップＳ１０３で“ＹＥＳ”となり、次のステップＳ１０５で、コンピュータ２０（図１）は、音Ｓｋにおける音高Pitch（Ｓｋ）と、音Ｐｊにおける音高Pitch（Ｐｊ）とを比較する。このステップＳ１０５で“ＹＥＳ”なら、コンピュータ２０は、次のステップＳ１０７で、時点ｔｊでの音符Ｓｋの重みW（Ｓｋ，ｔｊ）として、W（Ｓｋ-１，ｔｊ-１）＋１を設定する。そして、ステップＳ１０９で音数ｋをインクリメントして先のステップＳ１０３に戻る。ただし、この実施例では、ステップＳ１０７で加える値を「１」としたが、この値は「１」でなければならないというものではなく、他の適宜の値を加算するようにしてもよい。

【００５９】

ステップＳ１０５で“ＮＯ”なら、コンピュータ２０は、ステップＳ１１１で、時点ｔｊ－１での音符Ｓｋ－１の重みW（Ｓｋ-１，ｔｊ-１）が「１」より大きいかどうか、すなわちW（Ｓｋ-１，ｔｊ-１）－１＞０かどうか判断する。“ＹＥＳ”なら、次のステップＳ１１３で時点ｔｊでの音符Ｓｋの重みW（Ｓｋ，ｔｊ）としてW（Ｓｋ-１，ｔｊ-１）－１を設定する。ただし、ステップＳ１１１で“ＮＯ”なら、次のステップＳ１１５で、時点ｔｊでの音符Ｓｋの重みW（Ｓｋ，ｔｊ）として「０」を設定する。

【００６０】

このようなステップＳ１０１－Ｓ１１５をステップＳ１０３で“ＮＯ”が判断されるまで、つまり、すべての音について重みを評価するまで、繰り返し実行する。すべてのポジションで重みを評価した後は、原則的には、最大の重みを示す演奏位置を次の演奏位置として同定する。

【００６１】

すなわち、すべてのポジションについて重みWを評価したなら、ステップＳ１０３で“ＮＯ”となり、コンピュータ２０は、続くステップＳ１１７で、時点ｔｊでの演奏位置Ｓｉ＋１における重みが、時点ｔｊ－１での演奏位置Ｓｉにおける重みに「１」加えたものと等しいかどうか、つまりW（Ｓｉ＋１，ｔｊ）＝W（Ｓｉ，ｔｊ－１）＋１かどうか判断する。つまり、演奏位置が正しく更新されているかどうか判断する。ただし、この実施例では、ステップＳ１１７で加える値を「１」としたが、この値は「１」でなければならないというものではなく、他の適宜の所定値を加算するようにしてもよい。

【００６２】

ここで、このステップＳ１１７での条件判定について詳しく説明する。このステップＳ１１７では、演奏時点ｔｊにおける演奏位置の判定を試みている状態である。そして、演奏時点ｔｊ－１での演奏位置がＳｉである。したがって，W（Ｓｉ，ｔｊ－１）とは，前回の判定（ｔｊ－１）時点での判定で「ここが演奏位置だ」と判定した箇所の重みであ

る。そして、Ｗ（Ｓｉ＋１，ｔｊ）とは、楽譜上でＳｉの次の音Ｓｉ＋１の、今回（ｔｊ時点）での重み付け処理の結果得られた重みである。そして、Ｗ（Ｓｉ＋１，ｔｊ）の値がＷ（Ｓｉ，　ｔｊ－１）の値に「１」を加えた値になっていた場合（Ｗ（Ｓｉ，　ｔｊ－１）が図７のＳ１０７の処理によって１加算されていた場合）に、Ｓｉ＋１をｔｊ時点での演奏位置と判定するものである。

【００６３】

そして、このステップＳ１１７で“ＹＥＳ”なら、演奏位置が正しいのであるから、次の演奏位置Ｓｉ＋１をｔｊ時点の演奏箇所として、図7の演奏位置判定処理（図６のステップＳ１）を終了する。

【００６４】

(2)もし、Ｗ（Ｓｉ＋１，ｔｊ）≠Ｗ（Ｓｉ，ｔｊ－１）＋１であれば、ステップＳ１２１以降を実行して、演奏誤り（弾き直しを含む）への対応を行う。つまり、ステップＳ１１７で“ＮＯ”が判断されるということは、プリモは演奏を間違ったことを意味し、以下、正しい演奏位置を同定する処理を実行することになる。

【００６５】

まず、弾き直しについて、詳しく説明すると、「１」から「ｉ」の範囲にあって、かつ楽譜上の小節、フレーズ、段およびページ頭など、前述の実験（表１）で得られた「弾き直しが発生し易い個所」として予め楽譜データ上に指定されている個所すべての重みを、Ｗ（Ｓｉ＋１，ｔｊ）－ｍ（ｍは正の定数）とする。つまり、弾き直し位置マーカの設定されている演奏位置の重みをすべて最大より「ｍ」小さい値に設定する。ただし、その個所の元の重みがこの計算で得られた重みより大きい場合は、値を変更しない。なお、発明者等の実験においてｍの値は、経験的に「２」としている。ｍの値を小さくすると、弾き直しへの追従性が向上するが、一方で類似したパターンが繰り返し現れる楽曲の場合、軽微な誤り（音脱落など）で演奏個所の認識誤りを招く可能性が高くなる。つまり、ｍ個の音数だけ余裕を見て「弾き直し箇所」と思われる箇所からの音がｍ個連続して弾かれたとき初めて「弾き直し」として判定するようにしている。これによって、１音程度の音の抜けや挿入などの微少な誤りに過剰に反応しないようにしている。

【００６６】

ついで、挿入、音高誤り、脱落への対応について説明する。位置Ｓｉ-ｒからｓｉ+ｒまでの範囲の音すべてについて、重みをＷ（Ｓｉ+１，ｔｊ）と同じにする。つまり、ｔｊ-１時点での現在演奏位置の前後ｒ音ずつをｔｊ時点での演奏位置としての可能性を高く評価することにより、ｒ個までの挿入や音高誤り、脱落に対処している。なお、音数ｒは正の定数とし、実験においてはｒの値は、経験的に「２」としている。この値を大きくすると、より多数の音にわたる誤りに対処できるが、大きくし過ぎると可能性の範囲が広がりすぎ、逆に誤った一致箇所を見いだしてしまう危険性が高くなる。

【００６７】

(3)そして、重みＷ（Ｓｋ，ｔｊ）（ただし、１≦ｋ≦Ｎ）最大値をとる個所を現在の演奏位置とするのであるが、もし同じ重みの個所が複数ある場合は、以下の順に優先する。

(a) 重みＷ（Ｓｋ，ｔｊ）が最大値をとる箇所が唯一であるならば、そのときの重みＷ（Ｓｋ，ｔｊ）が最大値となる位置Ｓｋを現在の演奏位置とする。

(b) 重みＷ（Ｓｋ，ｔｊ）が最大値をとる箇所が複数個あるならば、現在の演奏位置が次のようにして決められる。

【００６８】

(i)Ｓｉに最も近い最大値を取る個所Ｓｋが唯一であるならばその個所Ｓｋを現在の演奏位置とする。

【００６９】

(ii)Ｓｉに最も近い最大値を取る個所が複数個ある場合、すなわちＳｉから等距離の位置Ｓｉ-ｄとｓｉ+ｄとに最大値が生じる場合は、Ｓｉ-ｄを現在の演奏位置とする。

【００７０】

このことを図７を参照して説明すると、図７のステップＳ１１７で“ＮＯ”となったときには、コンピュータ２０は、次のステップＳ１２１において、すべての弾き直し位置Ｓｈ（表１においてリターンのマーカが付与されている箇所）の重みとして、それらの重みを、最大よりｍだけ小さい値にするために、Ｗ（Ｓｉ＋１，ｔｊ）－ｍを設定する。ついで、ステップＳ１２３で、コンピュータ２０は、ステップＳ１２３で、位置Ｓｇ（＝ｉ-ｒ≦ｇ≦ｉ+ｒ）での重みＷ（Ｓｇ，ｔｊ）をＷ（Ｓｉ+1，ｔｊ）とする。つまり、そのときの演奏位置の前後の演奏位置の重みを等しくする。

【００７１】

そして、ステップＳ１２５において、重みＷ（Ｓｋ，ｔｊ）（ｋ＝１～Ｎ）が最大値をとる個所が複数あるかどうか判断する。“ＮＯ”の場合には、ステップＳ１２７において、コンピュータ２０は、重みＷ（Ｓｋ，ｔｊ）が最大値となる位置Ｓｋを現在ｔｊの演奏位置とする。

【００７２】

ステップＳ１２５において“ＹＥＳ”が判断されたときには、すなわち、重みＷ（Ｓｋ，ｔｊ）が最大値を取る個所が２以上存在するときには、続くステップＳ１２９において、コンピュータ２０は、位置Ｓｉに最も近い最大値の演奏個所Ｓｋが複数存在かするかどうか、判断する。ステップＳ１２９で“ＹＥＳ”のときには、コンピュータ２０は、ステップＳ１３１において、ｋ＜ｉの演奏個所Ｓｋを現在の演奏位置とする．ステップＳ１２９で“ＮＯ”なら、ステップＳ１３３で、コンピュータ２０は、位置Ｓｋを現在の演奏位置とする。

【００７３】

以上のアルゴリズムによって、従来から扱われてきた３種類の演奏誤りに加えて、初心者の演奏で頻発する「弾き直し」に対しても追従できる頑健な演奏追跡処理が実現される。

【００７４】

実験によれば、「弾き直しへの対応」を行った場合と行わなかった場合とでは、前者では、たとえば３音の誤認識の後正しい位置を検出した。これに対して、同じ曲でも後者の場合には、弾き直し後に正しい位置を検出するまでに１３音の誤認識が生じた。このように、この実施例の演奏位置判断アルゴリズムを採用した場合には、弾き直しへも十分実用的なレベルでの追従が可能となった。

【００７５】

次に、図８を参照して、演奏音高取得処理（図６のステップＳ３）を説明する。

【００７６】

演奏音高取得処理も、図7と同様に、プリモの打鍵というイベントによって駆動される。したがって、図８の最初のステップＳ３０１で、コンピュータ２０は、プリモのそのときの演奏位置Ｓｉのデータを受ける。そして、ステップＳ３０３における一定時間Ｔｗミリ秒の経過の後、ステップＳ３０５で、コンピュータ２０は、位置Ｓｉ＋１にセコンドが演奏すべき音が楽譜上にあるかどうか判断する。“ＮＯ”なら、ステップＳ３０７において、コンピュータ２０は、セコンドが演奏すべき音を「なし（０）」に設定する。一方、ステップＳ３０５で“ＹＥＳ”の判断をしたとき、つまり、位置Ｓｉ＋１においてセコンドが演奏すべき音があるとの判断をしたときには、次のステップＳ３０９において、コンピュータ２０は、そのときセコンドが演奏可能な音をたとえば図５のセコンドの欄に指定された音に設定する。

【００７７】

したがって、プリモが誤りなく演奏している場合、位置（ポジション）Ｓｉは１、２、３、４、…と順に演奏位置がこのルーチンに投入される。たとえばＳｉ＝９とプリモの演奏ポジションが通知された場合、Ｔｗミリ秒の間をとった後、ポジションＳｉ＋１=１０の位置にセコンドが演奏する音が指定されているかどうかを調べる。図５の例では、セコンドのポジション１０の位置には音が指定されていないので、ステップＳ３０５での判定は“ＮＯ”となる。したがって、セコンドが演奏可能な音は「なし」と設定される。この

状態では、セコンドが打鍵しても何も発音しない。

【００７８】

引き続きプリモが誤りなく演奏を続けた場合、Ｓｉ=１０でも同様の処理が行われ、セコンドは何も発音できない状態が続く。さらに、プリモの演奏位置がＳｉ＝１１となったとき、Ｔｗミリ秒の間を置いた後、ポジションＳｉ＋１（＝１２）のセコンドの演奏音を確認する。表１の例では、セコンドのポジション１２には「Ｃ５」の音が指定されているので、ステップＳ３０５の条件判定の結果は“ＹＥＳ”となり、ここでセコンドが演奏可能な音として、「Ｃ５」が設定され、発音可能状態になる。以上の手順で、プリモに誤りがない場合のセコンドの演奏可能音は切り替えられていく。

【００７９】

次に、セコンドが誤って発音すべき音を発音し忘れた場合について説明する。セコンドの演奏可能音は、セコンドによる打鍵がない限りは実際には発音されない。セコンドによる発音がされないままの状態で、プリモが次のポジションを演奏し、新しい演奏位置情報Ｓｉが図８のアルゴリズムに入力されると、セコンドが先の音を実際に発音したかどうかに拘らず、このアルゴリズムが実行され、セコンドが演奏すべき音も更新される。たとえばポジション３の位置でセコンドが演奏し、「Ｇ４」の音を発音したとする。その後、プリモがポジション４、５と演奏を進めると、セコンドの演奏可能音は「Ｂ４」に更新される。しかし、この状態でセコンドが誤って演奏しないままプリモがさらにポジション６、７と演奏を進めると、「Ｂ４」の音は発音されないまま、セコンドの演奏可能音は「Ｇ４」に切り替わる。したがって、この状態でセコンドが演奏すると、演奏し忘れた「Ｂ４」ではなく「Ｇ４」の音が発音する。このようにして、セコンドが誤っていくつかの演奏音を飛ばしてしまっても、セコンドの演奏可能音は常にプリモの演奏位置に対応したものとなるため、セコンドが途中で演奏を再開しても、飛ばした音ではなく、今弾くべき音が正しく発音される。

【００８０】

一方、セコンドが誤って本来演奏すべきでない箇所で余分に演奏した場合について説明する。たとえばポジション９でセコンドが演奏し「Ａ４」の音を発音したとする。この直後、プリモがポジション１０の音を演奏しないうちにセコンドが再び打鍵した場合を考える。もしこの再打鍵がプリモのポジション９の音（Ｃ５）の打鍵よりＴｗミリ秒以内であれば、再度「Ａ４」の音が発生する。一方、プリモのポジション９の音（Ｃ５）の打鍵よりＴｗミリ秒以上経過した場合は、次のポジション１０の音が発音される。ただし、現在の例ではセコンドのポジション１０には演奏可能音が指定されていないため、この場合は何も発音しないことになる。引き続き、プリモがポジション１０に演奏を進めた場合、セコンドの演奏可能音はポジション１１に指定されていない。ゆえに、ここでセコンドが打鍵しても、依然として何も発音されない。さらにプリモがポジション１１の音を発音し、Ｔｗミリ秒以上経過すると、セコンドの演奏可能音はようやくポジション１２の「Ｃ５」に設定される。以上のように、セコンドが何度余分な打鍵をしても、セコンドの演奏位置だけが一方的に先走ることは起こらない。余分な音の挿入が生じる可能性はあるが、それによってプリモとセコンドの演奏位置がずれることはない。

【００８１】

以上はプリモが演奏を誤らない場合の処理の流れの説明であるが、次にプリモが演奏を誤って、いくつかの音を飛ばした場合の処理を説明する。この場合も、基本的に処理の概要は変わらない。プリモが演奏を誤った場合は、演奏位置判定ルーチンから通知されるポジションデータＳｉが前回と不連続になるだけであり、そのことはこのアルゴリズムの処理に関係しない。たとえば、プリモがポジション１２の音を演奏した後、ポジション９に戻った場合を考える。この場合、演奏位置判定モジュールから通知される演奏位置情報は、演奏位置判定が理想的に動作すれば、Ｓｉ＝１２の次にＳｉ＝９が図７に示す音高取得のアルゴリズムに通知される。Ｓｉ＝１２が通知されてＴｗミリ秒経つと、セコンドの演奏可能音はポジション１３の「Ｂ４」となっている。この状態でプリモがポジション９の音を弾き、ほぼ同時にセコンドが打鍵すると、セコンドの演奏音はそのままポジション１

３の「Ｂ４」となってしまう。これは回避できない誤りである。しかし、演奏ポジションとしてＳｉ＝９が通知されているので、次のセコンドの演奏可能音はポジション１０の音となる。

【００８２】

この例ではセコンドのポジション１０の位置は空白なので、この場合は演奏可能音「なし」となる。以下、プリモが誤り無く演奏を続ければ、先に述べたのと同様に処理が進む。

【００８３】

ここで、ステップＳ３０３でのＴｗミリ秒の遅延の意味について説明する。楽譜を単純に用いるならば、たとえばプリモがＳｉ＝１３の音（すなわち「Ｄ５」）を弾いた瞬間に、セコンドの演奏可能音も同時にポジション１３の「Ｂ４」にすればよいことになる。しかし、現実にはプリモとセコンドの演奏タイミングが全くずれなしで一致することはありえない。プリモが若干先行する場合もあれば、セコンドが若干先行する場合もある。プリモが先行し、セコンドが遅れる場合は、上記のような同時切り替えを行っても問題なく、ポジション１３の位置でプリモが「Ｄ５」の音を弾いた直後にセコンドは「Ｂ４」の音を弾くことができる。しかし、セコンドが先行し、プリモが遅れた場合は、セコンドは切り替えられていない「前の音」を演奏してしまうことになる。つまり、ポジション１３の箇所でセコンドが先行してしまうと、セコンドは本来弾くべき「Ｂ４」の音ではなく、前の音である「Ｃ５」の音を弾いてしまうことになり、不都合である。

【００８４】

この問題を回避するために、Ｔｗミリ秒の間を設けて、その後にセコンドの演奏可能音を切り替えている。Ｔｗミリ秒の時間は、プリモとセコンドが同時発音すべき時に、セコンドがプリモよりも遅れる可能性のある時間幅の最大値で、現在のシステムでは２００ミリ秒としている。つまり、このアルゴリズムでは、たとえばプリモがＳｉ＝１１の「Ｅ５」の音を弾いたＴｗミリ秒後に、セコンドが弾ける音はポジション１２の「Ｃ５」の音に設定される。この状態で、プリモがＳｉ＝１２の「Ｅ５」を弾くと、その瞬間からＴｗミリ秒間は、セコンドは現在設定されている「Ｃ５」を弾ける状態が維持される。つまり、セコンドはプリモよりも遅延することＴｗミリ秒未満であれば、ここで弾くべき正しい音である「Ｃ５」を演奏できる。しかし、Ｔｗミリ秒以上セコンドによる発音が無い場合は、セコンドが今弾くべき音を「弾き忘れた」とシステムは見なし、次のポジション１３の「Ｂ４」の音を弾くべき音に設定し、次のプリモのＳｉ＝１３の発音に備える。このため、プリモがＳｉ＝１３の「Ｄ５」の音を演奏していない場合でも、セコンドの演奏音はすでにポジション１３の音になっているため、セコンドがプリモに若干先行して発音してしまっても、演奏すべきポジション１３の「Ｂ４」が正しく演奏される。以上のメカニズムによって、プリモとセコンドの演奏タイミングのずれによる演奏音の不一致の問題を吸収している。

【００８５】

続いて、図９に示す音高データ差替え処理（図６のステップＳ５）について詳しく説明する。この図９の処理は、セコンドによる打鍵というイベントで駆動され、最初のステップＳ５０１で、コンピュータ２０は、楽譜データを参照して、位置Ｓｉ＋１にプリモの演奏すべき音があるかどうか判断する。“ＹＥＳ”ならそのままこのルーチンを抜けるが、“ＮＯ”なら、次のステップＳ５０３において、コンピュータ２０は、セコンドが演奏可能な音を、楽譜データの位置Ｓｉ＋１のセコンドの欄に指定されている音に変更する。

【００８６】

具体的に説明すると、図９の音高データ差替えのアルゴリズムは、プリモに演奏すべき音がなく、セコンドにのみ演奏すべき音がある場合の処理を行っている。ここで扱っている例で言えば、ポジション１４～１６の場所がこの場合にあたる。プリモがポジション１３の「Ｄ５」を演奏してＴｗミリ秒経過すると、セコンドの演奏可能音はポジション１４の「Ａ４」になっている。この状態でセコンドが打鍵するとこの「Ａ４」の音が発音されると同時に、この図９のアルゴリズムが動作する。

【００８７】
プリモ側の演奏位置Ｓｉは１３のままなので、ポジションＳｉ＋１（＝１４）のプリモの楽譜データを調べ、プリモがポジション１４で演奏する音があるかどうかを確認する。この例では、プリモにはポジション１４で演奏するデータがないので、条件判定は“ＮＯ”となる。この場合、現在の演奏位置をＳｉ＝Ｓｉ＋１（＝１４）とし、セコンドの演奏可能音をＳｉ＋１（＝１５）の「Ｇ４」に設定する。この状態でセコンドが再度打鍵すると、「Ｇ４」の音が発音されると同時に、再度第２のアルゴリズムが動作し、以上と同様の処理を実施する。このようにして、プリモに演奏データが無い場合は、プリモではなくセコンドが演奏進行のイニシャティブを取って、プリモの演奏とは無関係に独自に演奏を進行させられるようになる。

【００８８】
なお、セコンドがまだポジション１６まで演奏し終わっていない状態でプリモがポジション１７の音「Ｅ５」を演奏した場合、演奏位置Ｓｉは一気にポジション１７に移動し、セコンドの未演奏音は無視されることになる。これは、演奏位置判定ルーチンが、セコンドの演奏位置を考慮しておらず、プリモの演奏データと楽譜データのみを参照しているためである。しかし、これはアルゴリズムの不備ではない。なぜならば、この実施例のシステムはあくまでプリモ主導のシステムであり、セコンドが主導となるのは限られた箇所で例外的に生じる状態であるからである。ゆえに、たとえセコンドのみが演奏すべき箇所であっても、プリモが演奏を再開した場合は、即座にプリモに主導権を渡す必要がある。この結果、プリモが間違って早く次の演奏位置に移動してしまっても、セコンドは正しくプリモの演奏位置に相当する演奏音を発音できる。

【００８９】
ところで、セコンドの打鍵があれば無条件にこの図９に示すアルゴリズムが起動されるので、実際にはプリモに演奏可能音が無い場合だけでなく、プリモに演奏可能音がある場合にもこの図９のアルゴリズムは起動する。しかし、その場合に、演奏位置Ｓｉ＋１にプリモが演奏すべき音があるかの条件判定が“ＹＥＳ”となるため、演奏ポジションＳｉもセコンドの演奏可能音も変更されない。つまりこの場合は結局何もしないでこのアルゴリズムは終わる。

【００９０】
以上のような、図８および図９に示す２つの簡単なアルゴリズムの非同期動作によって、プリモとセコンドの演奏位置が常に一致するように動作する。

【００９１】
このように、この発明の背景となる図１の合奏支援システム１０で発明者等が何組かのペアに頼んで行った実験によれば、いずれのペアも、この支援システムを利用することによって、すぐに合奏（連弾）が可能になった。

【００９２】
図１の合奏支援システム１０では、上で説明したように、プリモとセコンドとによる「連弾」を簡単に実現できる。そして、連弾の場合には、1台のキーボード型入力インタフェース（鍵盤）を２名の演奏者で共有することになる。図１の合奏支援システム１０では、プリモ側の演奏に対しては一切加工を加えず、そのまま楽音として出力する。つまり、第２奏者はどの鍵盤を押しても正しい音高を出力することができる仕組みであるが、連弾の場合には、第1奏者が使用する鍵盤の範囲を避けなければならない。さらに、第１奏者が使用せざる得ない鍵盤の範囲を第２奏者が打鍵してしまっては、第１奏者の邪魔をすることになるので、第２奏者が使用可能な鍵盤範囲を視覚的にも認識する必要がある。

【００９３】
そこで，図１０以下に説明するこの発明の一実施例の合奏支援システム１０では、事前にプリモが使用せざる得ない範囲の最低音から、セコンドが使用可能な範囲を事前に判定し、そのセコンドが使用可能な範囲においては、どの鍵盤を押しても、セコンドが各時点で演奏すべき正しい音高が出力できる仕組みに設定する。

【００９４】

図１０の実施例の合奏支援システム１０では、連弾を想定しているため、1台の入力インタフェース１２を使い、その上位音域をプリモ用の第１入力インタフェース１２ａとして使用し、下位音域をセコンド用の第２入力インタフェース１２ｂとして使用する。ただし、プリモとセコンドとが「連弾」するかどうかは、必須ではないので、図１の合奏支援システムと同様に、独立した２台の入力インタフェース１２および１４を使用して図１０の合奏支援システム１０を実装することも可能である。

【００９５】

図１０はこの発明の一実施例を示すブロック図であり、以下の点を除いて、図１実施例と同様である。

【００９６】

図１のシステムでは、２台の入力インタフェース１２および１４を用いたが、この実施例では、１台の鍵盤型入力インタフェース１２´を用いる。そして、その上位音域をプリモ領域１２ａとし、下位音域をセコンド領域１２ｂとする。そして、この１台の入力インタフェース１２´の鍵盤の奥側（図１０で上方）に、鍵盤の並び方向すなわち横方向に延びる領域表示器２４が設けられる。この領域表示器２４は、発光素子２６によって、プリモの使用可能範囲（またはセコンドの使用可能範囲）を目視可能に表示する。発光素子２６としては、発光ダイオード、ＥＬ（エレクトロルミネセンス）など任意の素子が利用可能である。また、この表示器２４では、発光素子２６が領域全体を面表示（バー表示）するようにしているが、入力装置１２の各鍵盤（キー）の位置に１つずつ発光素子を配置しておき、たとえばプリモの使用範囲の下端の位置に相当する１つの発光素子を点灯し、プリモとセコンドとの境目だけを表示するようにしてもよい。

【００９７】

さらに、この図１０実施例では、１台の入力インタフェース１２´からの演奏データ（ＭＩＤＩデータ）が一緒に、コンピュータ２０およびＭＩＤＩ音源１６に入力される。そして、コンピュータ２０は、入力された演奏データがプリモのものかセコンドのものかを判別し、セコンドのものであれば、先に説明したように、音高データだけを差し替えたＭＩＤＩデータをＭＩＤＩ音源１６に与える。ただし、コンピュータ２０はプリモの演奏データに加工を加えることはしない。

【００９８】

このように、プリモは、基本的に与えられた楽譜をそのまま演奏するため、その楽譜（楽譜データベース２２内の）を調査すれば、プリモが使用する音高の範囲を容易に決定できる。そこで、この実施例では、プリモが使用する最低音がどれかを求め、その音よりも低い範囲はプリモが使用することはないという前提で、プリモの最低音より低い範囲の鍵盤をセコンド用として割り当てる。こうすることにより、ユーザが自分で楽譜を調べてプリモとセコンドのそれぞれの領域を求めるという作業を行うことなく、楽譜データさえ入力すれば自動的にそれぞれの領域を確定できる。そのアルゴリズムを図１１を参照して説明する。このアルゴリズムでは、楽譜データを先頭から１行ずつ読み込みながらプリモの演奏音の最低音を求めるようにしている。

【００９９】

図１１の最初のステップＳ６０１では、コンピュータ２０は、メモリ２８（このメモリは、たとえばＲＡＭやＲＯＭを含む）の最低音レジスタLowestに初期値Ｃ９という文字列を代入する。

【０１００】

次のステップＳ６０３で、楽譜データベース２２の、そのとき演奏しようとしている曲の楽譜データに、読み込むべき行があるかどうか判断する。読み込むべき楽譜データがある場合には、楽譜データベース２２から楽譜データを１行読み込み、メモリ２６の適宜場所に形成または設定されている行データレジスタlineに読み込む。ただし、その楽譜データベース２２からその都度読み出す方法に代えて、その曲の楽譜データを一旦全部メモリ２６のＲＡＭ（図示せず）に読み込んでおき、そこから行データレジスタlineに代入するようにしてもよい。そうすれば、より高速に処理することができる。

【０１０１】
このときの行レジスタlineの内容は、たとえばつぎのようである。
【０１０２】
Ａ６，Ｃ７：Ａ４，５，Ｅ５：Ｃｏｍｍｅｎｔ
そして、次のステップＳ６０７では、コンピュータ２０は、行データレジスタlineに読み込んだ１行の楽譜データの最初の文字が「＊」かどうか判断する。この「＊」は、先に説明したように、曲を初心者が演奏する場合に戻りやすい位置を示すマーカであり、演奏すべき音のデータではないので、このステップＳ６０７で“ＹＥＳ”が判断されたときには、最初のステップＳ６０１に戻る。
【０１０３】
ステップＳ６０７で“ＮＯ”が判断されたとき、つまり、１行の最初の文字は演奏すべき音データであるときには、次のステップＳ６０９で、コンピュータ２０は、行データレジスタlineに代入している１行分の楽譜データの最初の「：（コロン）」までの文字列を、メモリ２６のプリモデータレジスタprimoに代入する。最初の「：」と次の「：」との間の文字列はセコンドの楽譜データであり、最後の「：」以降の文字列はコメントである。そして、コメントの内容は、演奏には反映されない。先の楽譜データの例でいえば、ステップＳ６０９のプリモデータレジスタprimoの内容は、「Ａ６，Ｃ７」となる。
【０１０４】
その後、ステップＳ６１１で、プリモデータレジスタprimoの文字列を「，（カンマ）」で分割し、各部分文字列を、メモリ２６内の配列レジスタpdata[]に格納するとともに、部分文字列の個数を個数レジスタpnに設定する。つまり、ステップＳ６１１では、レジスタprimoに格納されている文字列をさらに以下のように分解してpdataという配列変数に格納する。
pdata[0]←Ａ６，pdata[1]←Ｃ７，pn←２
つまり、配列レジスタpdataには行データレジスタlineのその時点でプリモが同時に演奏すべき楽音の音高を１つずつ分解して格納する。個数レジスタpnには、その時点でプリモが同時に発音する音の個数を格納する。上の例は、行データレジスタlineのこの箇所では、プリモは楽譜上では「Ａ６」という音高の音と、「Ｃ７」という音高の音の２つの音を同時発音することになっていることを示す。
【０１０５】
そして、ステップＳ６１３－Ｓ６２１の５つのステップでは、行データレジスタlineでの各時点でプリモが同時発音するすべての音について、行データレジスタlineの箇所以前に楽譜で指定されていた音よりも低い音がないかをチェックする。ステップＳ６１５での「pdata［i］＜Lowest?」の箇所がこのチェックに相当する。
【０１０６】
なお、この比較処理は、具体的に、次のようなルールに従う。
【０１０７】
(i) 文字列のうち２文字目または３文字目にある数値部分が小さいほうが小さい。
【０１０８】
(ii) 上記数値部分が同じ場合には、１文字目の文字の関係Ｃ＜Ｄ＜Ｅ＜Ｆ＜Ｇ＜Ａ＜Ｂに従って大小を判定する。
【０１０９】
(iii) 数値部分も、１文字目の文字も同じ場合
iii-0) いずれも２文字の場合は同じ大きさとする。
【０１１０】
iii-1) いずれか一方が３文字の場合には、３文字のほうの２文字目が「＋」であれば３文字のほうが大きく、３文字のほうの２文字目が「－」であれば３文字のほうが小さいものとする。
【０１１１】
iii-2) いずれも３文字の場合には、いずれか一方の２文字目が「＋」で、他方の２

文字目が「－」で「＋」のほうが大きく、いずれも「＋」または「－」であれば同じとする。

【０１１２】

初期状態では最低音レジスタLowestにはＣ９という非常に高い音（ピアノの音域を超えた高い音）が設定されている（Ｓ６０１）。したがって、上記の例で行データレジスタlineが楽譜の１行目の内容を読み込んだ状態であれば、ここでまず、配列レジスタpdata[0]内の「Ａ６」と最低音レジスタLowest内の「Ｃ９」とが比較され、どちらの音が低いかが判定される。この場合、「Ａ６」の方が低いので、ステップＳ６１７で、Lowest = pdata[i]として、最低音レジスタLowestの内容が書き換えられる。具体例では、最低音レジスタLowestは、現時点（ｉ＝０）において、「Ａ６」に書き換えられる。

【０１１３】

次のループでは、ステップＳ６１５で、配列レジスタpdata[1]のデータ「Ｃ７」と最低音レジスタLowestのデータ「Ａ６」とが比較される。このときは、最低音レジスタLowestの「Ａ６」の方が低いため、最低音レジスタLowestの書き換えはしない。

【０１１４】

このようなステップＳ６１３－Ｓ６２１によるループ処理を、次々に楽譜の行を読み込みつつ進めれば、最終的な最低音レジスタLowestには、プリモの楽譜に記載されたすべての楽音のうち、最も低い音高の値が格納されることになる。

【０１１５】

ステップＳ６０３で“ＮＯ”が判断たれた後、つまり、それ以上読み込むべき楽譜データがなくなったときに、ステップＳ６２３では、最低音レジスタLowestにある音高値が、これから連弾または合奏しようとしている曲のプリモの楽譜の最低音として決定される。

【０１１６】

そして、ステップＳ６２５で、コンピュータ２０は、表示信号を出力し、領域表示器２４で、上記最低音に相当する鍵（キー）よりも上（右側）がプリモの使用せざるを得ない範囲であることを知らせ、あるいは逆に、その最低音よりも下（左側）がセコンドの使用可能範囲であることを知らせる。このようにして、コンピュータ２０は、その曲を演奏する際のプリモが使用せざるを得ない範囲、およびセコンドの使用できる範囲を知ることができる。さらに、プリモにその領域表示器２４の表示を目視によって確認させることで、検索したプリモの最低音より低い位置のキーにタッチするのを防止できる。

【図面の簡単な説明】

【０１１７】

【図１】図１はこの発明の背景となる合奏支援システムを示す概略ブロック図である。

【図２】図２は楽譜の記述形式の一例を示す図解図である。

【図３】図３は楽譜の一例を示す図解図である。

【図４】図４は図３の楽譜を記述した記述形式を示す図解図である。

【図５】図５は図３の楽譜に従った楽譜データの一例を示す図解図である。

【図６】図６は図１実施例における演奏動作を全体的に示すフロー図である。

【図７】図７は、図６実施例における演奏位置判定処理の動作を示すフロー図である。

【図８】図８は、図６実施例における音高データ取得処理の動作を示すフロー図である。

【図９】図９は、図１実施例における音高データ差替え処理の動作を示すフロー図である。

【図１０】図１０はこの発明の一実施例の合奏支援システムを示す概略ブロック図である。

【図１１】図１１は図１０実施例におけるプリモ（第１奏者）の最低音の検索動作を示すフロー図である。

【符号の説明】

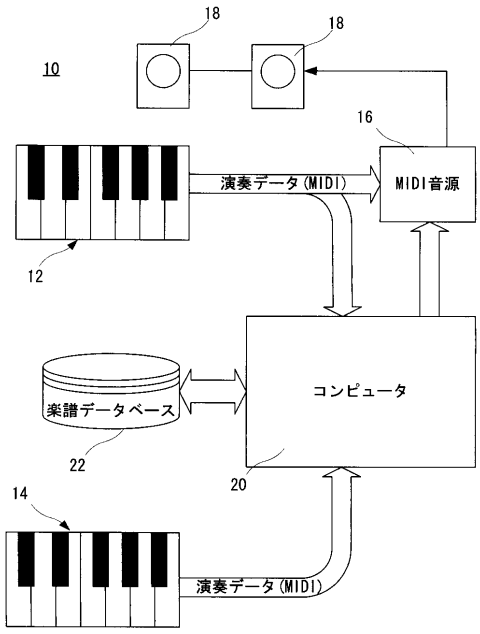
【０１１８】

１０　…合奏支援システム

１２　…プリモ用入力インタフェース

１４　…セコンド用入力インタフェース
１２´　…入力インタフェース
１６　…ＭＩＤＩ音源
１８　…スピーカ
２０　…コンピュータ
２２　…楽譜データベース
２４　…領域表示器
２６　…発光素子
２８　…メモリ

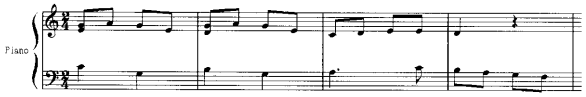
【図１】



【図２】

＊[BPRS]
プリモ 音名：セコンド音名：コメント
プリモ音名：セコンド音名：コメント
・・・
＊[BPRS]
プリモ音名：セコンド音名：コメント
・・・

【図３】



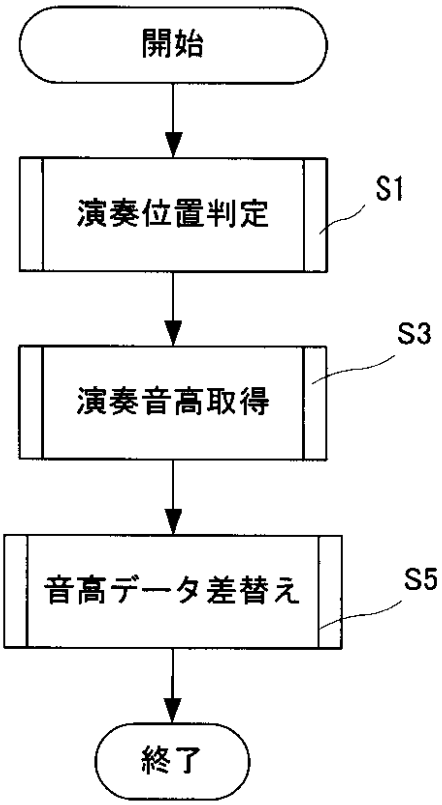
【図４】

＊BPR
G5,E5:C5:最初の音．プリモは2つの音を弾いている．
A5::セコンドは新たに演奏すべき音がない
G5:G4:
E5::
＊B
G5,D5:B4:2小節目．プリモは2つの音を弾いている．
A5::
＊S
G5:G4:このプリモの演奏者はここに戻り易い癖があるので，
直前に＊Sを埋め込んでいる．
E5::
＊BP
C5:A4:3小節目．
D5::
E5::
E5:C5:
＊B
D5:B4:
:A4: セコンドのみ演奏音がある
:G4:
:F4:

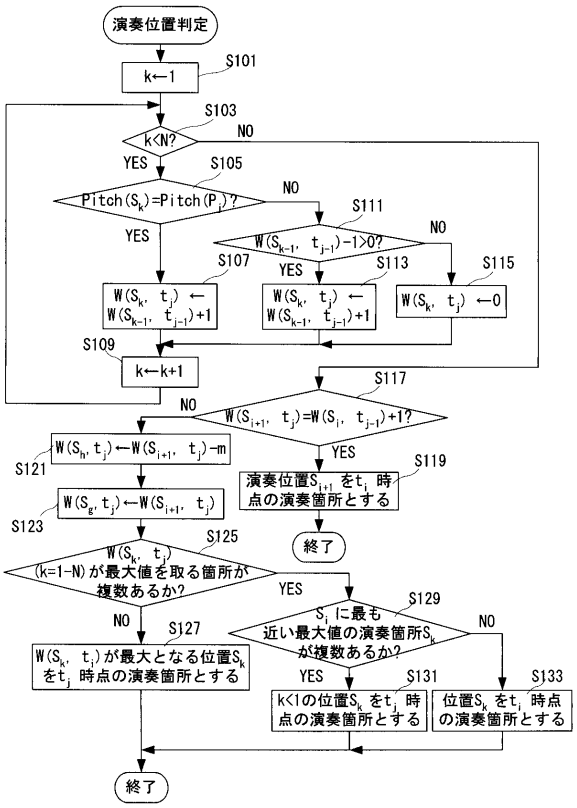
【図５】

ポジション	プリモ	セコンド	リターン
1	G5, E5	C5	*
2	A5		
3	G5	G4	
4	E5		
5	G5, D5	B4	*
6	A5		
7	G5	G4	*
8	E5		
9	C5	A4	*
10	D5		
11	E5		
12	E5	C5	
13	D5	B4	*
14		A4	
15		G4	
16		F4	
17	E5	C4	*

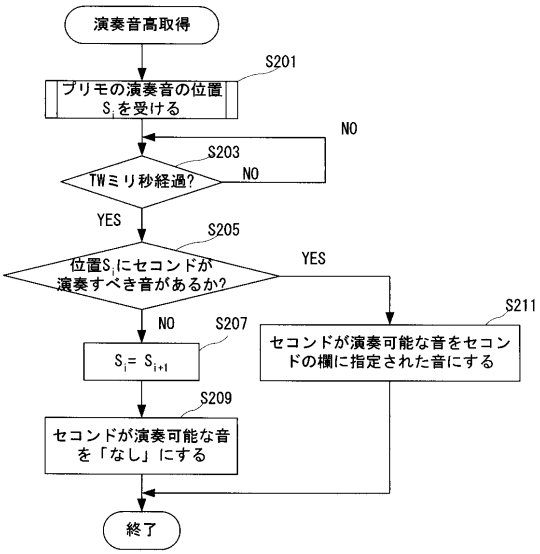
【図６】



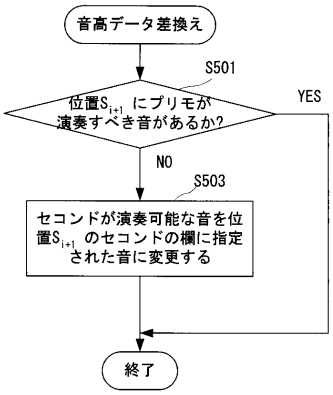
【図７】



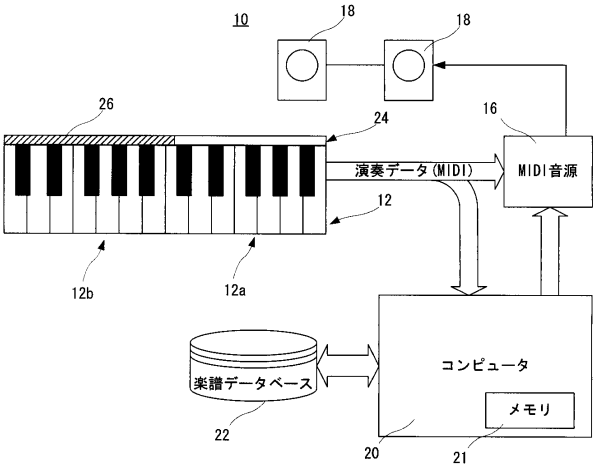
【図８】



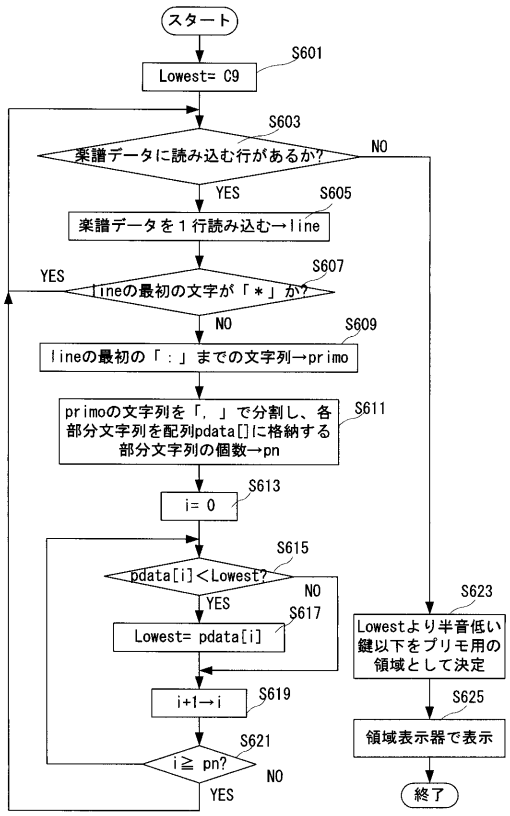
【図９】


音高データ差換え
S501
位置S_{i+1} にプリモが演奏すべき音があるか?
YES
NO
S503
セコンドが演奏可能な音を位置S_{i+1} のセコンドの欄に指定された音に変更する
終了

【図１０】


10
18
18
26
24
16
演奏データ(MIDI)
MIDI音源
12
12b
12a
楽譜データベース
コンピュータ
メモリ
22
20
21

【図１１】


スタート
S601
Lowest= C9
S603
楽譜データに読み込む行があるか?
NO
YES
S605
楽譜データを１行読み込む→line
S607
YES
lineの最初の文字が「＊」か?
NO
S609
lineの最初の「：」までの文字列→primo
S611
primoの文字列を「，」で分割し、各部分文字列を配列pdata[]に格納する
部分文字列の個数→pn
S613
i= 0
S615
pdata[i]＜Lowest?
NO
YES
S617
Lowest= pdata[i]
S619
i+1→i
S621
i≧ pn?
NO
YES
S623
Lowestより半音低い鍵以下をプリモ用の領域として決定
S625
領域表示器で表示
終了

フロントページの続き

(56)参考文献　特開平０８－３３５０８２（ＪＰ，Ａ）
特開平０８－２３４７４７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ１０Ｈ　　１／００