(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 600 22 343 T2** 2006.06.22

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 183 677 B1**
(21) Deutsches Aktenzeichen: **600 22 343.4**
(86) PCT-Aktenzeichen: **PCT/US00/13721**
(96) Europäisches Aktenzeichen: **00 936 067.8**
(87) PCT-Veröffentlichungs-Nr.: **WO 00/72303**
(86) PCT-Anmeldetag: **19.05.2000**
(87) Veröffentlichungstag der PCT-Anmeldung: **30.11.2000**
(97) Erstveröffentlichung durch das EPA: **06.03.2002**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **31.08.2005**
(47) Veröffentlichungstag im Patentblatt: **22.06.2006**

(51) Int Cl.[8]: ***G10H 5/00*** (2006.01)

(30) Unionspriorität:
135014 P **20.05.1999** **US**

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(73) Patentinhaber:
JohnMack LLC, San Carlos, Calif., US

(74) Vertreter:
Schoppe, Zimmermann, Stöckeler & Zinkler, 82049 Pullach

(72) Erfinder:
Jameson, John W., San Carlos, US; Ring, Mark B., Irvine, US

(54) Bezeichnung: **DURCH SPRACHE GESTEUERTES ELEKTRONISCHES MUSIKINSTRUMENT**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

HINTERGRUND DER ERFINDUNG

TECHNISCHES GEBIET

[0001] Die Erfindung bezieht sich auf Musikinstrumente. Insbesondere bezieht sich die Erfindung auf ein sprachgesteuertes elektronisches Musikinstrument.

BESCHREIBUNG DES STANDES DER TECHNIK

[0002] Musikinstrumente sind traditionell schwierig zu spielen, erfordern also eine beträchtliche Investition in Form von Zeit und in manchen Fällen auch Geld, bis man die grundlegenden Fertigkeiten des Bedienens dieses Instruments gelernt hat. Zusätzlich zu häufigen und oft mühsamen Übungsstunden ist üblicherweise Musikunterricht erforderlich, in dem die mechanischen Fertigkeiten zum Erzielen des richtigen musikalischen Ausdrucks, der diesem Instrument zugeordnet ist, z. B. Tonhöhe, Lautheit und Timbre, gelehrt werden. Ferner wird üblicherweise eine musikalische Sprache gelehrt, so dass der Benutzer in der Lage ist, das Instrument zu bedienen, um zuvor niedergeschriebene Stücke zu spielen.

[0003] Die Entwicklung von Musikinstrumenten verlief bisher relativ langsam, wobei sich über die letzten Jahrhunderte nur wenige neue Musikinstrumentprodukte durchgesetzt haben. Die Einführung der elektronikbezogenen Technologie hatte und hat jedoch einen beträchtlichen Einfluss auf die Entwicklung von Musikinstrumentprodukten. Der Musik-Synthesizer beispielsweise ließ zusammen mit der Klaviertastaturschnittstelle/-steuerung die Anzahl und Vielfalt von Instrumentenklängen, die seitens einer Person erzeugt werden können, die ein einziges Instrument – das des Klaviers oder Keyboards – zu spielen gelernt hat, immens ansteigen. Jedoch blieb das Erfordernis, dass jemand, der einen Synthesizer bedienen wollte, auch zumindest einige der Grundlagen des musikalischen Ausdrucks lernt, der mit dem Spielen eines Klaviers verbunden ist, bestehen.

[0004] Somit war für diejenigen Menschen, die in der Lage sein wollten, sich musikalisch auszudrücken, jedoch nicht gelernt hatten, ein Instrument zu spielen, oder die in der Lage sein wollten, viele Instrumentenklänge hervorzubringen, ohne jedes Instrument spielen zu lernen, immer noch eine beträchtliche Investition an Zeit erforderlich, um die Fertigkeiten zu erlernen, ohne Gewähr, dass sie jemals ein Leistungsniveau erlangen würden, das für sie akzeptabel sein würde.

[0005] In den U.S.-Patentschriften Nrn. 3,484,530 und 3,634,596 sind Systeme zum Erzeugen musikalischer Ausgaben aus einem Speicher offenbart, der aufgezeichnete Musiktöne enthält, die durch Eingaben einzelner Töne durch ein Mikrophon stimuliert werden können. Die in diesen Patentschriften offenbarten Systeme sind, so liest man, in der Lage, Tonhöhe, Einsatz, das Halten von Tönen und das Abklingen von Tönen sowie den Lautstärkepegel zu erfassen, und sind in der Lage, diese erfassten Eingaben auf den aufgezeichneten Ton, der gerade abgespielt wird, anzuwenden. In der Tat sind die Systeme Musikton-zu-Musikton-Umwandler, die schnell genug umgewandelt werden können, so dass seitens des Zuhörers oder des Spielers kein Zeitabstand wahrgenommen werden kann. Um diese Fähigkeiten zu erzielen, wurden jedoch ziemlich umständliche und teure elektronische und mechanische Mittel vorgeschlagen, die sich nicht für tragbare oder in der Hand zu haltende Instrumente eignen, sondern in erster Linie für größere Systeme gedacht sind.

[0006] Bei den in den obigen Patentschriften offenbarten Systemen ist der Speicher in der Lage, diskrete Töne der chromatischen Tonleiter zu enthalten und auf diskrete Eingabetöne derselben Tonhöhe anzusprechen. Das System ist analog zu einem Klaviaturinstrument, bei dem der Spieler lediglich aus einzelnen Tönen auswählen kann und durch Drücken der jeweiligen Taste einen Ton betätigt. Andere Musikinstrumente geben einem Spieler eine Auswahl von Tonhöhen zwischen Ganz- und Halbtonsprüngen vor. Beispielsweise kann eine Violine eine Tonhöhe erzeugen, die je nachdem, ob die Saite mit Bünden versehen ist, variabel ist, oder eine Schiebe-Posaune kann eine Tonhöhe hervorbringen, die zwischen Ganz- und Halbtonsprünge fällt. Beide diese Instrumente erzeugen eine kontinuierliches Tonhöhe-Frequenzspektrum. Jedoch sind derartige Systeme des Stands der Technik nicht in der Lage, ansprechend auf eine kontinuierlich variierende Tonhöhe am Eingang eine kontinuierlich variierende Tonhöhe am Ausgang zu liefern, noch sind sie bisher in der Lage, eine Tonklangfarbe zu erzeugen, die auf realistische Weise kopiert, was ein echtes Instrument als Funktion der Tonhöhe über die Bandbreite des Instruments tut, noch liefern sie eine Tonqualität oder eine Tonklangfarbe, die auf realistische Weise kopiert, was ein echtes Instrument als Funktion des Stärkegrades am Eingang eines Instruments tut.

[0007] Es wurden bereits eine Vielzahl anderer Verfahren vorgeschlagen, die menschliche Stimme zur Steuerung eines Synthesizers zu verwenden, womit also der einzige Mechanismus musikalischen Ausdrucks, den die meisten Menschen haben, genutzt wird. Praktisch jeder, der sprechen kann, hat die Fähigkeit, Parameter, die einen musikalischen Ausdruck darstellen, z. B. Tonhöhe und Lautheit, zu verändern. Ein derartiges Verfahren ist bei R. Rupert, U.S.-Patentschrift Nr. 4,463,650 (7. August 1984) beschrieben. Bei der Vorrichtung von Rupert sind echte instrumentale Töne in einem Speicher enthalten, wobei das System auf die Stimuli einer von ihm als solches bezeichneten „Mundmusik" anspricht, um spielbare Musikinstrumente zu erzeugen, die in Echtzeit auf die Mundmusikstimuli ansprechen. Siehe auch K. Obata, Input apparatus of electronic device for extracting pitch from input waveform signal, U.S.-Patentschrift Nr. 4,924,746 (15. Mai 1990).

[0008] Ishikawa, Sakata, Obara, Voice Recognition Interval Scoring System, europäische Patentschrift Nr. 142,935 (29. Mai 1985), unter Anerkennung der Ungenauigkeiten der Singstimme, „betrachtet das Bereitstellen einer Korrektureinrichtung zum mühelosen Korrigieren von in Partitur gesetzten Intervalldaten, und um das Intervall in einem Korrekturmodus zu korrigieren, indem Cursor an zu korrigierenden Abschnitten verschoben werden". Im Rahmen eines ähnlichen Versuchs, mit stimmlichen Ungenauigkeiten umzugehen, verwendet eine von M. Tsunoo u. a., U.S.-Patentschrift Nr. 3,999,456 (28. Dezember 1976), beschriebene Vorrichtung ein Sprachschlüsselsystem für ein sprachgesteuertes Musikinstrument, das den ausgegebenen Ton auf eine Musiktonleiter beschränkt. Die Schwierigkeit bei der Verwendung entweder der Ishikawa- oder der Tsunoo-Vorrichtung für nützliche Zwecke besteht darin, dass die meisten ungeübten Musiker nicht wissen, welche Tonleitern für verschiedene Stücke und Anwendungen geeignet sind. Aufgrund der Frustration des Benutzers, nicht in der Lage zu sein, bestimmte Töne, die er spielen möchte, zu erreichen, kann die Vorrichtung sogar eine Verschlechterung gegenüber dem nicht-verbesserten sprachgesteuerten Musik-Synthesizer darstellen.

[0009] Auf einem verwandten Gebiet ist das Konzept von „Musik-Minus-Eins" die Verwendung eines vordefinierten, üblicherweise vorab aufgezeichneten musikalischen Hintergrunds, um kontextbezogene Musik zu liefern, um die herum ein Musiker/Benutzer singt oder ein Instrument spielt, üblicherweise die führende Stimme. Dieses Konzept ermöglicht es dem Benutzer, voller klingende Musik zu machen, indem er eine Hauptstimme spielt, die anderen Stimmen jedoch von anderen Musikern spielen lässt. Vorteile einer solchen Erfahrung umfassen einen höheren Unterhaltungswert, Übungswert und eine Möglichkeit kreativen Ausdrucks.

[0010] M. Hoff, Entertainment and creative expression device for easily playing along to background music, U.S.-Patentschrift Nr. 4,771,671 (20. September 1988), offenbart eine Verbesserung des Konzepts Musik-Minus-Eins, wobei dem Musikinstrument, das bei dem sprachgesteuerten Musik-Synthesizer die Hauptstimme spielt, ein gewisses Maß an Intelligenz verliehen wird, in diesem Fall, um keinen Ton zu erzeugen, der bezogen auf die Hintergrundmusik dissonant oder misstönend klingt. Ferner offenbart Hoff eine Variation des sprachgesteuerten Musik-Synthesizers durch Verwendung einer Korrektur. Statt das Intervall bzw. den Tonabstand willkürlich zu korrigieren, wie in den Patentschriften von Tsunoo und Ishikawa vorgeschlagen wurde, passt diese Vorrichtung die Ausgabe des Musik-Synthesizers an eine Ausgabe an, die für den durchschnittlichen Zuhörer bezogen auf die vordefinierte Hintergrundmusik unbedingt gut klingt. Jedoch führt Hoff eine Korrektur der Tonhöhe lediglich im Kontext vorprogrammierter Begleitungen durch, und zwar unter Verwendung des Tonleitertons, der durch die Begleitung, die der erfassten Tonhöhe am nächsten kommt, vorgeschlagen wird. Hoff sieht keine Korrektur der Tonhöhe in der Abwesenheit einer Begleitung vor, beispielsweise der Möglichkeit, dass der Benutzer die für die Korrektur der Tonhöhe zu verwendende Tonleiter auswählt, oder der Möglichkeit, die derzeit erfasste Tonhöhe der Tonika jener Tonleiter zuzuweisen.

[0011] Man kennt verschiedene Lösungsansätze in Bezug auf den Prozess der Tonhöheerfassung selbst. Siehe z. B. M. Russ, Sound Synthesis and Sampling, Focal Press, 1996, S. 265, oder L. Rabiner u. a., A Comparative Performance Study of Several Pitch Detection Algorithms, IEEE Transactions on Acoustics, Speech, and Signal Processing, Bd. ASSP-24, Nr. 5, Oktober 1976, S. 399. Laut Russ sind die traditionellen allgemeinen Klassifizierungen für die Tonhöheerfassung a) Nulldurchgang, b) Autokorrelation, c) spektrale Interpretation. Zwei Autokorrelations-Lösungsansätze, die dem vorliegenden Lösungsansatz in gewisser Weise ähneln, sind z. B. S. Dame, Method and Device For Determining The Primary Pitch of A Music Signal, U.S.-Patentschrift Nr. 5,619,004 (8. April 1997), und M.J. Ross, H.L. Shaffer, A. Cohen, R. Freudberg und H.J. Manley, Average Magnitude Difference Function Pitch Extractor, IEEE Trans. on Acoustics, Speech, and Signal Processing, Bd. ASSP-22, Nr. 5 (Oktober 1974).

[0012] Es wurden auch bereits andere sprachbetätigte elektronische Instrumente beschrieben, z. B. D. B. Pitt, Voice Actuated Instrument, GB-A-1 393 542; G. Wagner, Didaktisches Elektronisches Musikinstrument, DE 30 09 864 A; und S. Takushima, T. Tokimoto und K. Kurosawa, Electronic Musical Instrument, U.S.-Patentschrift Nr. 4,633,748.

[0013] Ein bedeutender Nachteil der meisten derzeit bekannten Systeme, die eine Sprachsteuerung eines Musikinstruments ermöglichen, besteht darin, dass sie sperrige Gehäuse erfordern und in nichtvertrauten Formfaktoren präsentiert werden, d. h. als imposante technische Ausrüstungsstücke. Somit ist ein Benutzer nicht in der Lage, auf natürliche Weise mit derartigen Instrumenten eine Beziehung zu knüpfen. Solche Vorrichtungen verleihen einem dem Eindruck, dass man, statt ein Musikinstrument zu spielen, vielmehr ein Maschinenteil bedient, das in den meisten Fällen dem Bedienen eines Computers ähnelt. Allein diese Tatsache ist schon eine gute Erklärung für den fehlenden kommerziellen Erfolg und die fehlende Akzeptanz beim Verbraucher, die diese Vorrichtungen bisher begleiteten.

[0014] Es wäre vorteilhaft, ein sprachgesteuertes Musikinstrument in einem Formfaktor zu liefern, der das eigentliche Instrument, das das elektronische Instrument repräsentieren soll, am ehesten darstellt. Weiter wäre es vorteilhaft, wenn ein derartiger Formfaktor zur leichten Verwendung eines derartigen Instruments beitragen würde, indem er einem Benutzer eine einfache Bedienmethode liefert. Ferner wäre es vorteilhaft, eine rechentechnisch effiziente Tonhöheerfassungstechnik für ein sprachgesteuertes elektronisches Musikinstrument zu liefern, so dass ein Formfaktor einer verringerten Größe erzielt werden könnte.

ZUSAMMENFASSUNG DER ERFINDUNG

[0015] Die Erfindung liefert ein sprachgesteuertes Musikinstrument in einem Formfaktor, der das eigentliche Instrument, das das elektronische Instrument repräsentieren soll, am ehesten darstellt. Ein derartiger Formfaktor trägt zur leichten Verwendung eines derartigen Instruments bei, indem er einem Benutzer eine einfache Bedienmethode liefert. Ferner liefert die Erfindung eine rechentechnisch effiziente Tonhöheerfassungstechnik für ein sprachgesteuertes elektronisches Musikinstrument zu liefern, so dass ein Formfaktor einer verringerten Größe erzielt werden könnte.

[0016] Die in dem vorliegenden Dokument beschriebene Vorrichtung ist ein elektronisches, sprachgesteuertes Musikinstrument. Sie ist im Wesentlichen ein elektronisches Kazoo. Der Spieler summt in das Mundstück, und die Vorrichtung imitiert den Klang eines Musikinstruments, dessen Tonhöhe und Lautstärke sich ansprechend auf die Stimme des Spielers ändern.

[0017] Dem Spieler wird der Eindruck verliehen, das eigentliche Instrument zu spielen und es auf intime Weise mit den feinen Nuancen seiner Stimme zu steuern. Bedeutend ist, dass die Vorrichtung kompakt und in sich abgeschlossen ist und durch den Benutzer mit einem einfachen Satz von Steuerungen bedient wird. Auf eine solche Weise überwindet die Erfindung viele der Hindernisse auf dem Weg zu einer Akzeptanz derartiger elektronischer Instrumente, wie sie im Stand der Technik gelehrt wurden. Das heißt, dass die Vorrichtung während des Spielens einfach zu bedienen und zu halten ist. Da die Vorrichtung in sich abgeschlossen ist, ein geringes Gewicht aufweist und vollständig integriert ist, liegen keine Drähte frei bzw. sind keine Verbindungen zwischen verschiedenen Komponenten eines Systems herzustellen, was sowohl den Genuss der Vorrichtung als auch das Empfinden, dass die Vorrichtung ein elektronischer Ersatz für das eigentliche Instrument ist, das sie physisch darstellt, schmälern würde. Da die Vorrichtung in einer zweckgebundenen Form, z. B. als Blasinstrument, vorgesehen ist, wird der Benutzer in die musikalische Erfahrung hineingezogen, statt durch die Verwendung eines Mikrophons abgelenkt zu werden. Somit liegt ein stimmliches Bedienen der Vorrichtung sehr stark nahe, dass das eigentliche Instrument, das die Vorrichtung darstellt, gespielt werde, und erzeugt den Eindruck, dass der Benutzer tatsächlich ein Instrument spielt. Indem ferner die kontraintuitive Maßnahme ergriffen wird, die Fähigkeit des Benutzers, die Bedienung der Vorrichtung zu verändern, stark einzuschränken, ist die Benutzerschnittstelle bedeutend vereinfacht. Dies wiederum verleiht der Vorrichtung die Form und Bedienung des eigentlichen Instruments, so dass der Benutzer das Gefühl haben mag, er spiele das Instrument, obwohl er vielleicht nicht die musikalischen Fähigkeiten besitzt, das eigentliche Instrument zu bedienen. Da die Vorrichtung ein einzigartiges Tonhöheerfassungsschema verwendet, das sowohl rechentechnisch effizient als auch für eine integrierte Vorrichtung, z. B. das hierin offenbarte sprachgesteuerte elektronische Musikinstrument, gut geeignet ist, ist es möglich, sowohl eine kompakte, in sich abgeschlossene Vorrichtung als auch, bedeutenderweise, eine Vorrichtung zu liefern, die ein hohes Maß an Musikalität liefert, wodurch der Eindruck, dass der Benutzer tatsächlich ein Musikinstrument spielt, weiter verstärkt wird.

[0018] Das Instrument kann im Prinzip eine beliebige Musik erzeugende Klangquelle sein: eine Trompete, eine Posaune, ein Saxophon, eine Oboe, ein Fagott, eine Klarinette, eine Flöte, ein Klavier, eine elektrische Gitarre, eine Stimme, eine Pfeife, d. h. praktisch jede beliebige Klangquelle.

[0019] In seiner einfachsten Konfiguration ähnelt das Instrument einer Art Blasinstrument und wird der Zweckmäßigkeit halber in dem gesamten vorliegenden Dokument als HumHorn (dt. etwa: Summhorn) be-

zeichnet. Jedoch kann die Gestalt und das Erscheinungsbild des Instruments seitens des Herstellers so gestaltet werden, dass diese auf den Klang eines beliebigen traditionellen Instruments abgestimmt sind, falls gewünscht; oder seine Gestalt kann völlig neuartig sein. Die funktionalen Anforderungen an die physische Ausgestaltung des HumHorns sind lediglich folgende:

- dass es in der Hand zu halten ist;
- dass es ein Mundstück hat – wo die Stimme des Spielers eintritt;
- dass es ein oder mehr Lautsprecher aufweist – wo der Klang erzeugt wird; und
- dass es einen Körper hat – wo die Elektronik und die Batterien aufbewahrt werden und wo mit dem Finger betätigte Steuerungen platziert werden können.

[0020] Drei Haupt-Softwarekomponenten des HumHorns sind das Frequenzerfassungsmodul, das Lautheitnachverfolgungsmodul und das Toneinsatzmodul.

[0021] Das Frequenzerfassungsmodul (FDM) identifiziert die Frequenz der Stimme des Spielers. Es tut dies, indem es die ankommende Klangwelle analysiert und Muster von wiederkehrenden Formen findet. Dieses Verfahren ist rechentechnisch äußerst effizient und eine neuartige Kombination einer Autokorrelation und einer Nulldurchgang- oder Spitzenbasierten Tonhöheerfassung. Das gewählte Instrument wird bei der durch das FDM ermittelten Tonhöhe oder bei einer Abweichung bzw. einem Versatz von dieser Tonhöhe, je nachdem, was der Spieler wünscht, synthetisiert.

[0022] Die Lautheitnachverfolgungskomponente misst die Lautheit der Stimme des Spielers, und diese Informationen werden anschließend verwendet, um die Lautstärke des synthetisierten Klangs einzustellen.

[0023] Das Toneinsatzmodul erfasst abrupte Änderungen der Lautheit der Stimme des Spielers. Diese Komponente hilft zu entscheiden, wann das synthetisierte Instrument einen neuen Ton beginnen sollte.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0024] Fig. 1 ist eine schematische Darstellung eines sprachgesteuerten elektronischen Musikinstruments gemäß der Erfindung;

[0025] Fig. 2 ist eine detailliertere schematische Darstellung eines sprachgesteuerten elektronischen Musikinstruments gemäß der Erfindung; und

[0026] Fig. 3 ist ein Blockdiagramm, das die Komponenten eines sprachgesteuerten Musikinstruments gemäß der Erfindung zeigt;

[0027] Fig. 4 ist ein Prozessfluss, der ein Signalanalysemodul gemäß der Erfindung zeigt;

[0028] Fig. 5 ist ein Signalverlauf, der ein Beispiel eines Eingangssignals zeigt;

[0029] Fig. 6 ist ein Blockdiagramm, das die Komponenten eines Frequenzerfassungsmoduls gemäß der Erfindung zeigt;

[0030] Fig. 7 ist eine Pseudocode-Auflistung, die einen Test zum Bestimmen, ob ein aktueller Abtastwert eine Spitze ist, zeigt, gemäß der Erfindung;

[0031] Fig. 8 ist eine Pseudocode-Auflistung für eine Routine, die über verschiedene Wellensegmente iteriert, gemäß der Erfindung;

[0032] Fig. 9 ist eine Pseudocode-Auflistung für eine Routine, die zwei Wellensegmente vergleicht, gemäß der Erfindung;

[0033] Fig. 10 ist eine Pseudocode-Auflistung für eine Routine, die die Frequenz berechnet und korrigiert, gemäß der Erfindung;

[0034] Fig. 11 ist ein Signalverlauf, der zwei mehrdeutige Wellen zeigt;

[0035] Fig. 12 ist ein Signalverlauf, der eine Lautheitnachverfolgung unter Verwendung von Frequenzinformationen zeigt, gemäß der Erfindung;

[0036] Fig. 13 ist ein Spiel- und Einsatzentscheidungsmodul gemäß der Erfindung;

[0037] Fig. 14 ist eine Pseudocode-Auflistung für ein Spiel- und Einsatzentscheidungsmodul gemäß der Erfindung;

[0038] Fig. 15 ist eine Pseudocode-Auflistung für eine Routine zum Schätzen der Frequenzzuverlässigkeit, gemäß der Erfindung;

[0039] Fig. 16 ist ein Blockdiagramm, das die Komponenten eines Klangsynthetisierungsmoduls zeigt, gemäß der Erfindung;

[0040] Fig. 17 ist eine Pseudocode-Auflistung für einen Nachrichtenprozessor gemäß der Erfindung;

[0041] Fig. 18 ist eine Pseudocode-Auflistung für eine Routine, den nächstliegenden Ton in einem spezifizierten Modus zu finden, gemäß der Erfindung;

[0042] Fig. 19 ist eine C++-Code-Auflistung für einen ersten Abschnitt eines Konstrukteurs für eine GetPitch-Klasse gemäß der Erfindung;

[0043] Fig. 20 ist eine C++-Code-Auflistung für einen zweiten Abschnitt des Konstrukteurs für eine GetPitch-Klasse gemäß der Erfindung;

[0044] Fig. 21 ist eine C++-Code-Auflistung für einen ersten Abschnitt einer GetPitch-Routine gemäß der Erfindung;

[0045] Fig. 22 ist eine C++-Code-Auflistung für einen zweiten Abschnitt der GetPitch-Routine gemäß der Erfindung;

[0046] Fig. 23 ist eine C++-Code-Auflistung für einen dritten Abschnitt der GetPitch-Routine gemäß der Erfindung;

[0047] Fig. 24 ist eine C++-Code-Auflistung für einen vierten Abschnitt der GetPitch-Routine gemäß der Erfindung;

[0048] Fig. 25 ist eine C++-Code-Auflistung für einen ersten Abschnitt einer Code match-Routine (Code Übereinstimmung-Routine) gemäß der Erfindung;

[0049] Fig. 26 ist eine C++-Code-Auflistung für einen zweiten Abschnitt der Code match-Routine (Code Übereinstimmung-Routine) gemäß der Erfindung;

[0050] Fig. 27 ist eine C++-Code-Auflistung für eine Frequenzkorrekturroutine gemäß der Erfindung;

[0051] Fig. 28 ist eine C++-Code-Auflistung für eine Lautheitnachverfolgungsroutine gemäß der Erfindung;

[0052] Fig. 29 ist eine C++-Code-Auflistung für eine Lautheiteinsatzerfassungsroutine gemäß der Erfindung;

[0053] Fig. 30 ist eine C++-Code-Auflistung für einen ersten Abschnitt einer Frequenz-„Güte"-Einschätzung-Routine gemäß der Erfindung;

[0054] Fig. 31 ist eine C++-Code-Auflistung für einen zweiten Abschnitt der Frequenz-„Güte"-Einschätzung-Routine gemäß der Erfindung;

[0055] Fig. 32 ist eine C++-Code-Auflistung für eine Einsatz- und Spiel-Bedingung-Routine gemäß der Erfindung; und

[0056] Fig. 33 ist ein schematisches Diagramm, das eine Architektur eines neuralen Netzwerks zum Bestimmen der Wahrscheinlichkeit einer Wellenformübereinstimmung zeigt.

AUSFÜHRLICHE BESCHREIBUNG DER ERFINDUNG

Glossar

[0057] Für die Zwecke der hierin erfolgenden Erörterung weisen die folgenden Begriffe die folgende Bedeutung auf:

ADC:	Analog/Digital-Wandler, wandelt analoge Spannungen in digitale Abtastwerte um.
Amplitude:	1) Wenn auf einen Punkt in dem Signal Bezug genommen wird, ist sie die Entfernung dieses Punktes von der Mittellinie. Wenn die Mittellinie null ist, wie in dem gesamten vorliegenden Dokument angenommen wird, so ist die Amplitude der Absolutwert des Abtastwerts an diesem Punkt. 2) Wenn auf eine Welle oder ein Wellensegment Bezug genommen wird, ist sie der Absolutwert des am weitesten von der Mittellinie entfernten Abtastwerts.
Einsatz:	Der Anfangsteil oder Beginn eines Tons.
DAC:	Digital/Analog-Wandler, wandelt digitale Abtastwerte in analoge Spannungen um.
F, F(t):	siehe Grundfrequenz.
Gefiltertes Signal (gefilterter Abtastwert):	Das Eingangssignal (der Eingangsabtastwert), nachdem es das Bandpassfilter durchlaufen hat. Bei diesem Dokument geht man davon aus, dass alle Signale und Abtastwerte gefiltert sind, wenn nicht explizit etwas anderes angegeben ist.
Grundfrequenz, F(t):	Die Frequenz der niedrigsten in dem Signal vorliegenden Tonhöhe. Sie ist die Frequenz, die als die Tonhöhe erkennbar ist, die gesungen oder gesummt wird.
Grundwelle:	Jegliche Welle in dem Signal, deren Länge die Grundwellenlänge ist. Sie ist die längste wiederholte Welle.
Grundwellenform:	Die längste unteilbare, wiederholte Wellenform
Grundwellenlänge, W(t):	Die (zeitliche) Länge der längsten sich wiederholenden Welle im Eingang. Sie entspricht der wahrgenommenen Tonhöhe des Eingangssignals bei einem gegebenen Zeitschritt und ist die Inverse der Grundfrequenz: d. h. W(t) = 1/F(t).
Halbschritt:	Dasselbe wie Halbton.
L, L(t), Lautheit:	Dieser Wert, der die Lautstärke des Eingangssignals darstellt, entspricht dem Wahrnehmungsbegriff von Lautstärke oder Lautheit. Es ist schwer, ihn mathematisch zu beschreiben, da er eine Mischung aus mehreren Faktoren ist. Er ist sehr stark auf die Amplitude einer Welle bezogen, ist in gewissem Maße jedoch auch auf die in der Welle enthaltenen Frequenzen bezogen. Bei diesem Dokument bezieht sich Lautheit immer auf das von dem Spieler empfangene Eingangssignal. Siehe Lautstärke.
Lautheitssprung:	Ein plötzlicher Anstieg der Lautheit der Stimme des Spielers. Er ist ein guter Hinweis darauf, dass der Spieler möchte, dass das Instrument einen Einsatz erzeugt.
MIDI:	Die digitale Musikinstrument-Schnittstelle (Musical Instrument Digital Interface), ein Standardprotokoll, das für eine digitale Kommunikation zwischen Musikinstrumenten und Computern verwendet wird. Sie wird von nahezu allen elektronischen Tastaturen und digitalen Musiksynthesesystemen verwendet.

Modus, Musikmodus bzw. Tonart, Musiktonart:	Ein Satz Halbtöne, die dem Satz aller elf möglichen Halbtöne in einer Oktave entnommen sind, die Musiker verwenden, um Töne in einem Musikabschnitt zu charakterisieren. Beispiele sind: Durtonart, Molltonart, Bluestonart und viele andere esoterische Tonarten, z. B. dorisch, phrygisch, Ganzton-, fünfstufig usw.
Ton:	1) Der Klang einer durch ein Instrument gespielten Note. Ein Ton beginnt mit einem Einsatz, auf den eine Halteperiode und anschließend ein Abklingen folgt. 2) Eine diskrete, ganzzahl-wertige Tonhöhe; d. h. der Index einer ganzzahl-wertigen Tonhöhe auf einer linearen Musiktonleiter, z. B. in einem musikalischen Notensystem oder auf einer Klaviertastatur, wo ein Intervall von 1 in der Tonleiter einem Halbschritt entspricht. Somit sind zwei Töne, die eine Oktave auseinander liegen, in der Tonleiter durch eine Differenz von 12 getrennt. Siehe Tonhöhe.
Toneinsatz:	Der Anfangsteil oder Beginn eines Tons.
Wahrgenommene Tonhöhe:	Die meisten Klänge, einschließlich der durch Stimmen und Musikinstrumente erzeugten, bestehen aus vielen verschiedenen Frequenzen und weisen viele überlappende Tonhöhen auf. Allgemein nimmt das menschliche Ohr zwei verschiedene Musikklänge als äquivalent wahr, wenn sie die niedrigste Frequenz (oder Grundfrequenz) gemeinsam haben, d. h. sie scheinen dieselbe Tonhöhe zu haben, ungeachtet aller anderer vorhandenen Frequenzen.
Tonhöhe:	Die Position einer Hörfrequenz auf einer linearen Musiktonleiter, z. B. in einem musikalischen Notensystem oder auf einer Klaviertastatur, wo zwei Tonhöhen, die eine Oktave auseinander liegen, durch eine konstante Anzahl von Schritten, z. B. 12, getrennt sind. Im Gegensatz dazu weisen zwei Frequenzen, die eine Oktave auseinander liegen, ein feststehendes Verhältnis, d. h. 2, auf. Die lineare Tonhöhenskala ist für das menschliche Verständnis eingängiger als die logarithmische Skala von Frequenzen. In dem vorliegenden Dokument wird Tonhöhe als kontinuierlicher Wert verstanden, der auf einer beliebigen Position auf der Musiktonleiter liegen kann. Ein Ton dagegen ist ein diskreter Ganzzahlwert.
R:	siehe Abtastrate.
SAM:	Das Signalanalysemodul, das das Eingangssignal Abtastwert um Abtastwert analysiert und Tonhöhe-, Lautstärke- und Einsatzwerte sowie einen Wert ausgibt, der anzeigt, ob ein Ton spielen sollte oder nicht.
Abtastwertrate, Abtastrate:	Die Frequenz, mit R bezeichnet, bei der der ADC (oder DAC) analoge Spannungen in digitale Abtastwerte (oder digitale Abtastwerte in analoge Spannungen) umwandelt. Übliche Abtastraten für Hörsignale betragen 8.000 Hz, 11.025 Hz, 22.050 Hz, 44.100 Hz und 48.000 Hz; je höher die Abtastrate, desto höher die Wiedergabetreue. Wenn nichts anderes angegeben ist, beziehen sich „Abtastwertrate" und „Abtastrate" auf das Abtasten des Eingangssignals.
Abtastwert, Abtastwert(t), s(t):	Ein Zeit-indexierter, digital codierter Wert, der die Höhe der Signalwelle zu einem bestimmten Moment darstellt. Eingangsabtastwerte werden durch den ADC an das SAM geleitet. Der Eingangsabtastwert zu einem Zeitschritt t wird mit s(t) bezeichnet. Ausgangsabtastwerte werden von dem SSM an den DAC geleitet. Wenn nichts anderes angegeben ist, bezieht sich „Abtastwert" auf den Eingangsabtastwert.

Abtasten:	Der Vorgang des Umwandelns eines Eingangssignals in eine Serie von digital codierten Zahlen. Dieser Vorgang wird durch den ADC durchgeführt.
Halbton:	Ein Halbschritt, das kürzeste Intervall bzw. der kürzeste Tonabstand, das bzw. der in der standardmäßigen westeuropäischen Musik verwendet wird. Er ist das Intervall zwischen benachbarten Tasten auf einem Klavier und zwischen benachbarten Tönen in einem Notensystem. In einer Oktave gibt es zwölf gleichmäßig beabstandete Halbtöne. Zwei Töne, die einen Halbton auseinander liegen, weisen somit ein Frequenzverhältnis von $2^{1/12}$ auf.
Signal:	Eine Sequenz von analogen Werten, die sich über die Zeit ändern. Ein Hörsignal sieht, wenn es auf einen Graphen aufgetragen ist, wie eine Welle aus. In jedem Moment weist das Eingangssignal einen bestimmten Wert auf, der in eine digitale Zahl umgewandelt werden kann, die die Spannung des Signals an diesem Punkt darstellt. Der Umwandlungsvorgang wird als Abtasten bezeichnet. Er wird durch den ADC durchgeführt. Desgleichen wird durch den DAC ein Ausgangssignal erzeugt, wenn er über die Zeit eine Serie von digital codierten Abtastwerten empfängt. Wenn nichts anderes angegeben ist, bezieht sich „Signal" auf das Eingangssignal.
Starke Spitze:	Eine Spitze in dem Eingangssignal, die die Kriterien bezüglich STARKER Spitzen des Kästchens **61** in Fig. 6 erfüllt. Siehe beiliegenden Text.
Lautstärke:	In dem vorliegenden Dokument bezieht sich Lautstärke immer darauf, wie laut das durch das HumHorn erzeugte Ausgangssignal ist. Siehe Lautheit.
W, W(t):	siehe Grundwellenlänge.
Wellenform:	Die Kontur, einschließlich der Größe, der Anzahl und der Position der Spitzen und Täler in einem gegebenen Wellensegment.
Wellenlänge:	Die zeitliche Länge zwischen dem Anfang einer Welle und dem Anfang der folgenden Welle. Sie ist die Inverse der Frequenz.
Schwache Spitze:	Eine Spitze in dem Eingangssignal, die die Kriterien in Bezug auf SCHWACHE Spitzen des Kästchens **61** in Fig. 6 erfüllt. Siehe beiliegenden Text.

Erörterung

[0058] Die Erfindung liefert ein sprachgesteuertes Musikinstrument in einem Formfaktor, der das eigentliche Instrument, das das elektronische Instrument repräsentieren soll, am ehesten darstellt. Ein derartiger Formfaktor trägt zur leichten Verwendung eines derartigen Instruments bei, indem er einem Benutzer eine einfache Bedienmethode liefert. Ferner liefert die Erfindung eine rechentechnisch effiziente Tonhöheerfassungstechnik für ein sprachgesteuertes elektronisches Musikinstrument zu liefern, so dass ein Formfaktor einer verringerten Größe erzielt werden könnte.

[0059] Die in dem vorliegenden Dokument beschriebene Vorrichtung ist ein elektronisches, sprachgesteuertes Musikinstrument. Sie ist im Wesentlichen ein elektronisches Kazoo. Der Spieler summt in das Mundstück, und die Vorrichtung imitiert den Klang eines Musikinstruments, dessen Tonhöhe und Lautstärke sich ansprechend auf die Stimme des Spielers ändern.

[0060] Dem Spieler wird der Eindruck verliehen, das eigentliche Instrument zu spielen und es auf intime Weise mit den feinen Nuancen seiner Stimme zu steuern. Bedeutend ist, dass die Vorrichtung kompakt und in sich abgeschlossen ist und durch den Benutzer mit einem einfachen Satz von Steuerungen bedient wird. Auf eine solche Weise überwindet die Erfindung viele der Hindernisse auf dem Weg zu einer Akzeptanz derartiger elektronischer Instrumente, wie sie im Stand der Technik gelehrt wurden. Das heißt, dass die Vorrichtung während des Spielens einfach zu bedienen und zu halten ist. Da die Vorrichtung in sich abgeschlossen ist, ein geringes

Gewicht aufweist und vollständig integriert ist, liegen keine Drähte frei bzw. sind keine Verbindungen zwischen verschiedenen Komponenten eines Systems herzustellen, was sowohl den Genuss der Vorrichtung als auch das Empfinden, dass die Vorrichtung ein elektronischer Ersatz für das eigentliche Instrument ist, das sie physisch darstellt, schmälern würde. Da die Vorrichtung in einer zweckgebundenen Form, z. B. als Blasinstrument, vorgesehen ist, wird der Benutzer in die musikalische Erfahrung hineingezogen, statt durch die Verwendung eines Mikrophons abgelenkt zu werden. Somit liegt ein stimmliches Bedienen der Vorrichtung sehr stark nahe, dass das eigentliche Instrument, das die Vorrichtung darstellt, gespielt werde, und erzeugt den Eindruck, dass der Benutzer tatsächlich ein Instrument spielt. Indem ferner die kontraintuitive Maßnahme ergriffen wird, die Fähigkeit des Benutzers, die Bedienung der Vorrichtung zu verändern, stark einzuschränken, ist die Benutzerschnittstelle bedeutend vereinfacht. Dies wiederum verleiht der Vorrichtung die Form und Bedienung des eigentlichen Instruments, so dass der Benutzer das Gefühl haben mag, er spiele das Instrument, obwohl er vielleicht nicht die musikalischen Fähigkeiten besitzt, das eigentliche Instrument zu bedienen. Da die Vorrichtung ein einzigartiges Tonhöheerfassungsschema verwendet, das sowohl rechentechnisch effizient als auch für eine integrierte Vorrichtung, z. B. das hierin offenbarte sprachgesteuerte elektronische Musikinstrument, gut geeignet ist, ist es möglich, sowohl eine kompakte, in sich abgeschlossene Vorrichtung als auch, bedeutenderweise, eine Vorrichtung zu liefern, die ein hohes Maß an Musikalität liefert, wodurch der Eindruck, dass der Benutzer tatsächlich ein Musikinstrument spielt, weiter verstärkt wird.

[0061] Wesentliche Aspekte der Erfindung umfassen:

- Tragbarkeit – da es kompakt und in sich abgeschlossen ist, kann das hierin offenbarte Instrument leicht transportiert werden, während es gespielt wird oder nicht gespielt wird, beispielsweise in einer Marschkapelle.
- Zugänglichkeit – aufgrund seiner einfachen Schnittstelle kann das hierin offenbarte Instrument von Körperbehinderten, z. B. teilweise oder vollständig gelähmten Menschen, verwendet werden.
- Simuliertes musikalisches Können – dies beschreibt den Vorgang des Spielens des Instruments.

[0062] Das HumHorn imitiert die Erfahrung des Spielens bzw. Darbietens eines eigentlichen Musikinstruments, einschließlich der visuellen, taktilen und auditiven Qualitäten der Erfahrung, und einschließlich der feinabgestimmten auditiven Steuerung eines Instruments, die bisher nur Musikern oblag, die auf dem Gebiet eines Musikinstruments geschult waren, und ferner einschließlich aller persönlichen, psychologischen und sozialen Vorteile, die der Vorgang des Bedienens eines eigentlichen Musikinstruments, ob solo oder zusammen mit anderen Interpreten, ob vor Publikum oder allein, mit sich bringt.

[0063] Das Instrument kann im Prinzip eine beliebige Musik erzeugende Klangquelle sein: eine Trompete, eine Posaune, eine Klarinette, eine Flöte, ein Klavier, eine elektrische Gitarre, eine Stimme, eine Pfeife, sogar ein Stimmenchor, d. h. praktisch jede beliebige Klangquelle.

[0064] In seiner einfachsten Konfiguration ähnelt das Instrument einer Art Blasinstrument und wird der Zweckmäßigkeit halber in dem gesamten vorliegenden Dokument als HumHorn (dt. etwa: Summhorn) bezeichnet. Jedoch kann die Gestalt und das Erscheinungsbild des Instruments seitens des Herstellers so gestaltet werden, dass diese auf den Klang eines beliebigen traditionellen Instruments abgestimmt sind, falls gewünscht; oder seine Gestalt kann völlig neuartig sein. Die funktionalen Anforderungen an die physische Ausgestaltung des HumHorns sind lediglich folgende:

- dass es in der Hand zu halten ist;
- dass es ein Mundstück hat – wo die Stimme des Spielers eintritt;
- dass es ein oder mehr Lautsprecher aufweist – wo der Klang erzeugt wird; und
- dass es einen Körper hat – wo die Elektronik und die Batterien aufbewahrt werden und wo mit dem Finger betätigte Steuerungen platziert werden können.

[0065] Drei Haupt-Softwarekomponenten des HumHorns sind das Frequenzerfassungsmodul, das Lautheitnachverfolgungsmodul und das Toneinsatzmodul.

[0066] Das Frequenzerfassungsmodul (FDM) identifiziert die Frequenz der Stimme des Spielers. Es tut dies, indem es die ankommende Klangwelle analysiert und Muster von wiederkehrenden Formen findet. Dieses Verfahren ist rechentechnisch äußerst effizient und eine neuartige Kombination einer Autokorrelation und einer Nulldurchgang- oder Spitzenbasierten Tonhöheerfassung. Das gewählte Instrument wird bei der durch das FDM ermittelten Tonhöhe oder bei einem Versatz von dieser Tonhöhe, je nachdem, was der Spieler wünscht, synthetisiert. Man kennt verschiedene Lösungsansätze bezüglich des Prozesses der Tonhöheerfassung selbst. Wie oben erörtert wurde, offenbart Russ, dass die traditionellen allgemeinen Klassifizierungen für eine Tonhöheerfassung a) Nulldurchgang, b) Autokorrelation, c) spektrale Interpretation lauten. Jedoch ist der vor-

liegende Lösungsansatz rechentechnisch viel effizienter, da die Wellenformen nur über Zeitspannen verglichen (korreliert) werden, die durch ein Unterscheiden zwischen Wellencharakteristika wie z. B. Spitzen oder Nulldurchgänge begrenzt sind, und nicht über Zeitspannen, die durch willkürliche Abtastpunkte begrenzt sind. Für den letzteren Fall ist eine viel größere Anzahl von Korrelationsberechnungen erforderlich. Der vorliegende Lösungsansatz nutzt einfach die Tatsache, dass Wellen durch Unterscheidungsmerkmale wie z. B. Spitzen oder Nulldurchgänge segmentiert werden können. Den Klassifizierungen von Russ zufolge ist der vorliegende Lösungsansatz eine neuartige Kombination der Klassifizierungen (a) und (b) und liefert die Genauigkeit einer Autokorrelation zusammen mit der rechentechnischen Effizienz der Nulldurchgangsverfahren. Als Verbesserung der Autokorrelation berücksichtigt der vorliegende Lösungsansatz außerdem eine Tonhöheänderung über die Zeit, indem er die verglichenen Wellen auf dieselbe Länge streckt oder schrumpfen lässt, bevor die Korrelation durchgeführt wird.

[0067] Die Lautheitnachverfolgungskomponente misst die Lautheit der Stimme des Spielers, und diese Informationen werden anschließend verwendet, um die Lautstärke des synthetisierten Klangs einzustellen.

[0068] Das Toneinsatzmodul erfasst abrupte Änderungen der Lautheit der Stimme des Spielers. Diese Komponente hilft zu entscheiden, wann das synthetisierte Instrument einen neuen Ton beginnen sollte.

[0069] Das HumHorn ist ein in der Hand zu haltender Musik-Synthesizer, dessen Ausgabe durch die menschliche Stimme gesteuert wird. Fig. 1 stellt die Funktionalität des Hum-Horns schematisch dar. Der Spieler **10** singt oder summt in das Mundstück **14** des Instruments **12**. Ansprechend darauf erzeugt das HumHorn an dem Ausgang **13** den Klang eines Musikinstruments, das in Bezug sowohl auf die Tonhöhe als auch die Lautstärke genau den Nuancen der Stimme des Spielers folgt. Der Spieler kann wählen, welches Instrument das HumHorn imitieren soll, und ihm wird der Eindruck vermittelt, nur durch Singen das gewählte Instrument zu spielen.

[0070] Wesentlich für die Erfindung ist die Tatsache, dass der Formfaktor der Vorrichtung ein Musikinstrument ist und dass alle Komponenten der Vorrichtung in dem Instrument selbst enthalten sind. Auf diese Weise wird dem Benutzer nahezu vollständig der Eindruck vermittelt, tatsächlich ein Instrument zu spielen und nicht einen Computer oder eine andere elektronische Vorrichtung zu bedienen. Man geht davon aus, dass allein diese Tatsache bedeutend genug ist, um die Technophobie zu überwinden, die viele Menschen einschüchtert, wenn sie sich neuen Technologien gegenüber sehen. Indem die Erfindung also in ein vertrautes Gehäuse, z. B. ein Blasinstrument oder ein anderes hinreichend bekanntes Musikinstrument, platziert wird, wird eine psychologische Barriere überwunden, was ermöglicht, dass die Vorrichtung durch eine breitere Gruppe von Personen genutzt wird. Da sie außerdem klein ist, ein geringes Gewicht aufweist, kompakt und einfach zu bedienen ist, wird eine physische Barriere überwunden, was körperbehinderten Menschen die Fähigkeit verleiht, ein Musikinstrument zu spielen. Indem außerdem die Metapher eines Musikinstruments geliefert wird, wird dem Benutzer und dem Publikum der Eindruck vermittelt, dass tatsächlich ein richtiges Musikinstrument gespielt wird. Dies bedeutet, dass die durch die Vorrichtung erzeugten Klänge mit dem Instrument, dem sie ähnelt, übereinstimmen, wie durch den Benutzer erwartet wird.

[0071] Das HumHorn selbst kann jedem bekannten oder neuartigen Instrument ähneln. Eine mögliche Konfiguration ist in Fig. 2 gezeigt. Bei diesem Modell führt das Mundstück **5** direkt zu dem Mikrophon **9**. Der Lautsprecher befindet sich in einem Doppelkegel-Abschnitt **3**, von dem ein Kanal durch das zentrale Gehäuse **11** zu einem Glockenabschnitt **7** führt, wo der Klang übertragen wird. Somit verleiht das Gehäuse dem erzeugten Klang eine akustische Qualität. Die Elektronik und die Batterien sind in dem zentralen Gehäuse enthalten, das ferner mehrere mit den Fingern betätigte Steuerungen aufweist: sowohl Druckknöpfe **1b** als auch Auswahlschalter **1a**. Diese Steuerungen ermöglichen es dem Spieler, Synthesizer-Parameter, z. B. Instrumentenauswahl, Lautstärke oder Oktave, zu verändern.

[0072] Die logische Struktur des HumHorns ist in Fig. 3 schematisch dargestellt. Das Mikrophon **30** sendet ein analoges Signal an einen Analog/Digital-Wandler (ADC) **31**, der das Signal bei einer feststehenden Frequenz, vorzugsweise 22.050 Hz, abtastet. Der ADC wandelt zu jedem Zeitpunkt immer nur einen Abtastwert um und sendet ihn an ein Bandpassfilter **32** (das das Signal glättet, indem es Frequenzen entfernt, die zu hoch oder zu niedrig sind). Jeder gefilterte Abtastwert wird anschließend an das Signalanalysemodul (SAM) **33** gesendet, wo er in dem Kontext der vorhergehenden Abtastwerte analysiert wird. Nach dem Analysieren des Abtastwerts leitet das SAM die folgenden Informationen an den Synthesizer **38** weiter:

- ob der Synthesizer einen Ton spielen soll oder nicht, und falls ja:
- die aktuelle Frequenz,
- die aktuelle Lautstärke (Lautheit);
- ob die Bedingungen für einen neuen Toneinsatz erfasst wurden.

[0073] Neben diesen Informationen von dem SAM empfängt der Synthesizer auch eine Eingabe von den mit den Fingern betätigten Steuerungen **37**. Diese Steuerwerte können eine Vielzahl von Synthesizer-Parametern modifizieren, einschließlich (aber nicht ausschließlich) der folgenden:

- des zu imitierenden aktuellen Instruments (Klangquelle);
- der Abweichung von der Stimme des Spielers, d. h. ob der synthetisierte Ton mit derselben Tonhöhe gespielt werden soll, mit der er gesungen wird, oder ob er bei einem festgelegten Intervall über oder unter dieser Tonhöhe gespielt werden soll;
- ob der Synthesizer immer die durch das SAM erfasste genaue Frequenz spielen sollte (kontinuierliche Tonhöhenachverfolgung) oder ob er stattdessen den dieser Frequenz am nächsten gelegenen Ton in einem festgelegten Musikmodus spielen soll (diskrete Tonhöhenachverfolgung);
- des Musikmodus, der für eine diskrete Tonhöhenachverfolgung zu verwenden ist, z. B. chromatisch, Dur, Moll, Blues; und
- ob die aktuelle Tonhöhe die Tonika (erster Ton) in dem gegebenen Musikmodus ist.

[0074] Gemäß allen zugeleiteten Informationen wird seitens des Synthesizers ein Ausgangsabtastwert erzeugt, und dieser Ausgangsabtastwert wird einem Digital/Analog-Wandler (DAC) **34** zugeführt. Aus einem Strom von digitalen Ausgangsabtastwerten, die er empfängt, erzeugt der DAC ein analoges Ausgangssignal. Dieses Signal wird an einen Verstärker **35** gesendet, bevor es durch den Lautsprecher **36** übertragen wird.

[0075] Der Rest des vorliegenden Dokuments liefert eine ausführliche Erörterung der oben umrissenen Komponenten. Die Softwarekomponenten (die aus Fig. 3) werden zuerst beschrieben. Hardwarekomponenten werden als zweites beschrieben.

Softwarekomponenten

[0076] Die nachfolgende Erörterung beschreibt zunächst das Filter. Als Nächstes beschreibt die Erörterung die Kern-Softwarekomponente, das SAM, das aus drei Teilmodulen besteht: dem Frequenzerfassungsmodul (FDM), dem Spiel- und Einsatzentscheidungsmodul (PADM) und dem Lautheitnachverfolgungsmodul (LTM). Danach beschreibt die Erörterung das Klangsynthetisierungsmodul (SSM).

Das Filter

[0077] Das Filter nimmt das rohe Eingangssignal direkt von dem ADC und filtert es, Abtastwert um Abtastwert, digital. Das digitale Filter darf frühere Abtastwerte betrachten, es kann jedoch keine zukünftigen Werte sehen. Das Filter glättet die Rohdaten, wobei es gezackte Spitzen entfernt, die üblicherweise nicht direkt auf die von dem Spieler beabsichtigte Tonhöhe bezogen sind. Es wird ein einfaches Bandpassfilter dritter Ordnung verwendet. Das Filter weist eine niedrige Sperrung von 200 Hz und eine hohe Sperrung von 300 Hz auf. Ein bevorzugtes Filter ist bei W. Press, B. Flannery, S. Teukolsky, W. Vetterling, Numerical Recipes in C, S. 456 – 460, Cambridge University Press (1988), beschrieben. Von diesem Punkt an beziehen sich alle Bezugnahmen auf das Signal, auf Abtastwerte und auf Wellen immer auf die gefilterten Werte, und alle Graphen zeigen gefilterte Werte an. Die rohen, ungefilterten Werte werden als unzugänglich angenommen. Wenn also das digitale Filter durch eine analoge Schaltungsanordnung ersetzt werden sollte, müsste der Rest dieses Dokuments nicht geändert werden.

Das Signalanalysemodul (SAM)

[0078] Das Signalanalysemodul (SAM) nimmt den aktuellen Abtastwert als Eingang **40** und erzeugt als Ausgang die vier oben beschriebenen Informationsposten: Ton ein/aus **41**, Frequenz **42**, Lautheit **43** und Einsatz **44**. Die Beziehung zwischen den drei Teilmodulen des SAM ist in Fig. 4 schematisch dargestellt. Der Eingangsabtastwert steht allen drei Teilmodulen zur Verfügung. Das FDM **45** berechnet sowohl die Frequenz des Eingangssignals als auch eine Maßzahl der Zuverlässigkeit dieser Berechnung. Erstere wird an das SSM **38** weitergesendet (Fig. 3), wohingegen die Letztere durch das PADM **46** verwendet wird. Das PADM nutzt auch den durch das LTM **47** berechneten Lautheitswert. Diese Komponenten und ihre Beziehungen werden in den folgenden Abschnitten beschrieben.

Das Frequenzerfassungsmodul (FDM)

[0079] Das Frequenzerfassungsmodul (FDM) analysiert das Eingangssignal, um die Grundfrequenz zu entdecken. Es tut dies, indem es in den Formen der ankommenden Wellen nach Mustern sucht. Die Grundwellenlänge ist die größte wiederholte Form.

[0080] Fig. 5 zeigt eine Welle, die einer Welle ähnelt, die eine menschliche Stimme nach einem Bandpassfiltern erzeugen könnte. Die horizontale Achse stellt die Zeit dar; Punkte auf der rechten Seite erfolgen nach Punkten auf der linken Seite. Die vertikale Achse stellt die Signalspannung dar. Punkte oberhalb der mittleren horizontalen Linie weisen eine positive Spannung auf. Punkte unterhalb dieser Linie weisen eine negative Spannung auf. Der ADC wandelt diese Spannungen in digitale Abtastwerte um. Bei dem bevorzugten 8-Bit-ADC fallen die Abtastwerte in den Bereich ±128 (ein 16-Bit-ADC erzeugt Werte im Bereich ±32.768). Je größer der durchschnittliche Betrag von Abtastwerten einer Welle, desto lauter ist sie.

[0081] Die Spitzen sind gegebene Markierungen, 1 – 17, die die Reihenfolge, in denen sie auftreten, darstellen. Der Begriff Spitze wird verwendet, um sowohl auf hohe (ungeradzahlige) als auch auf niedrige (geradzahlige) Spitzen zu verweisen. Der Zeitpunkt, zu dem eine Spitze auftritt, wird als t_p bezeichnet, wobei p die Nummer der Spitze ist; z. B. wird der Zeitpunkt, zu dem Spitze 1 auftrat, t_1 geschrieben, usw. Die Welle erstreckt sich von t_1 bis t_{17} und besteht aus einer Grundwelle, die viermal wiederholt wird: t_1 bis t_5, t_5 bis t_9, t_9 bis t_{13} und t_{13} bis t_{17}. Die Dauer oder Länge dieser Welle (z . B . $t_{13} – t_9$) ist die Grundwellenlänge und ist umgekehrt proportional zur Grundfrequenz.

$$F(t) = 1/W(t) \quad (1)$$

[0082] wobei W(t) die Grundwellenlänge und F(t) die Grundfrequenz ist. Das FDM findet diese Grundwellenlänge, indem es die längste unteilbare, wiederholte Wellenform – die Grundwellenform – findet. Sie ist unteilbar, wenn sie nicht selbst gänzlich aus einer wiederholten Wellenform besteht. Beispielsweise passt in Fig. 5 die Wellenform von t_{13} bis t_{17} zu der von t_9 bis t_{13} und ist die Grundwellenform. Obwohl das Segment von t_9 bis t_{17} zu dem Segment von t_1 bis t_9 passt, ist es nicht die Grundwellenform, da es in die zwei kleineren passenden Segmente teilbar ist.

[0083] Diese Technik – Identifizieren der Grundfrequenz durch Finden der Grundwellenform – arbeitet zugunsten des Hum-Horns, da das Eingangssignal die menschliche Stimme ist und bestimmte Eigenschaften dieses Eingangssignals im Voraus bekannt sind. Erstens kann die Stimme lediglich eine bestimmte Bandbreite von Frequenzen und Wellenlängen hervorbringen. Somit können Wellen, die länger oder kürzer sind als diese Bandbreite, ignoriert werden, wodurch der Verarbeitungsaufwand auf einem vernünftigen Maß gehalten wird. Zweitens kann die menschliche Stimme effektiv bandpassgefiltert werden, und der sich ergebende Signalverlauf ist glatt und verhält sich wohl (siehe unten). In diesem Kontext ist eine sich wohl verhaltende Welle eine Welle, bei der die Grundwelle lediglich eine geringe Anzahl von Spitzen – üblicherweise nicht mehr als vier oder fünf – überspannt. Dies hilft auch den Suchaufwand zu begrenzen.

[0084] Das FDM findet die Grundwellenform, indem es in jüngerer Zeit aufgetretene Segmente der Eingangswelle auf der Suche nach der größten wiederholten Form vergleicht. Die Effizienz des Formabstimmverfahrens des FDM ist auf eine fundamentale Einsicht zurückzuführen: Da die Grundwellenform immer durch Spitzen begrenzt wird, kann bei der Suche nach passenden Wellenformen stark gespart werden, indem lediglich Wellensegmente, die durch Spitzen begrenzt sind, verglichen werden. Aus diesem Grund werden Frequenzberechnungen nur dann durchgeführt, wenn eine neue Spitze erfasst wird. Da die Grundwelle üblicherweise nicht mehr als vier oder fünf Spitzen aufweist, beträgt die Anzahl der Vergleiche allgemein nicht mehr als etwa 25 (wie in Kürze erläutert wird), und der Durchschnitt ist viel geringer als diese Zahl.

[0085] Der gesamte Vorgang der Frequenzerfassung ist in Fig. 6 dargestellt. Mit jedem neuen Abtastwert **60** wird ein Test durchgeführt, ob dieser Abtastwert klar eine neue Spitze **61** darstellt. Falls dies nicht der Fall ist, wird die Frequenz unverändert auf ihrem vorherigen Wert belassen **65**. Wenn der Abtastwert jedoch klar eine neue Spitze darstellt, wird eine sich über in jüngerer Zeit aufgetretene vorherige Spitzen erstreckende Suche durchgeführt, um Wellensegmente zu finden, deren Formen vielleicht zueinander passen **62**. Wenn keine gefunden werden, wird die Frequenz wiederum unverändert belassen **65**. Wenn es mögliche Übereinstimmungen gibt, werden sie ausführlicher verglichen **63**, und die beste Übereinstimmung wird verwendet, um die neue Frequenz zu berechnen **64**.

Test bezüglich Spitze

[0086] Das in Fig. 6 mit **61** bezifferte Kästchen testet, ob der aktuelle Abtastwert eine Spitze darstellt. Der Test misst die erste und die zweite Ableitung der Welle an dem aktuellen Punkt. Es gibt drei mögliche Ergebnisse: STARK, SCHWACH und KEINE. Ein Pseudocode für diesen Test ist in Fig. 7 gezeigt. Zeilen 1 – 3 definieren, dass t die aktuelle Zeit ist, dass Abtastwert(t) der Wert des Eingangsabtastwerts bei dem aktuellen Zeitschritt ist und dass Neigung(t) die Neigung bei dem aktuellen Zeitschritt misst. Es gibt zwei Arten von Spitzen: hohe

Spitzen (diejenigen, die sich nach unten krümmen – Zeile 5) und niedrige Spitzen (diejenigen, die sich nach oben krümmen – Zeile 6). Die Krümmung ist gleich dem Betrag der zweiten Ableitung an dem Abtastpunkt gesetzt (Zeile 7). Die vertikalen Striche „||" stellen einen Absolutwert dar. Wenn der Abtastwert weder eine hohe Spitze noch eine niedrige Spitze ist, dann liegt keine Spitze vor (Zeile 8, 9). Wenn der Betrag des Abtastwerts niedriger ist als eine Schwelle (Zeile 10) oder wenn der Betrag der zweiten Ableitung zu niedrig ist (Zeile 12), dann existiert die Spitze, ist aber schwach (Zeile 11 oder 13). Andernfalls ist der Punkt eine starke Spitze (Zeile 15). Eine Wellenformsuche wird nur durchgeführt, wenn eine starke Spitze vorliegt.

[0087] Man beachte, dass der Test in Zeile 10 nur dazu dient, die Anzahl von Spitzen, die für eine Frequenzschätzung verwendet werden (starke Spitzen), zu verringern, und somit die Rechenlast insgesamt zu verringern. Dieses bestimmte Verfahren zum Aussortieren der Spitzen ist ziemlich willkürlich. Der Test der Zeile 10 könnte ganz und gar eliminiert werden, um die Rate der Frequenzschätzung zu Lasten eines zusätzlichen Rechenaufwands zu erhöhen.

Iteration über einzelne Wellensegmentpaare

[0088] Wenn der aktuelle Abtastwert eine starke Spitze ist, so zählt das in Fig. 6 mit **62** bezifferte Kästchen (Iteriere über einzelne Wellensegmentpaare) alle Paare von in jüngerer Zeit aufgetretenen Wellensegmenten auf und sendet sie an das mit **63** bezifferte Kästchen, damit sie verglichen werden. Als Beispiel dessen, was bei Kästchen **62** zu tun ist, sei erneut auf Fig. 5 Bezug genommen. Man nehme an, dass die Spitze bei t_{17} in Kästchen **61** gerade erfasst wurde. Sie wird nun als Endpunkt für das zweite von zwei zu vergleichenden Segmenten verwendet. Das erste Segment, Welle1, beginnt an einer Spitze, die vorübergehend als Start markiert ist, und endet an einer späteren Spitze, die vorübergehend als Aufteilen markiert ist. Das zweite Segment, Welle2, beginnt an der Aufteilen-Spitze und endet an der soeben im Kästchen **61** erfassten Spitze, die als Aktuell bezeichnet wird. Anfänglich ist Aufteilen die vorletzte Spitze, und Start ist die unmittelbar vorhergehende Spitze. Dann beginnt ein iterativer Prozess, mittels dessen die Markierungen Start und Aufteilen von einer Spitze zur nächsten zeitlich nach hinten bewegt werden. Jedes Mal, wenn eine Markierung bewegt wird, werden die neuen Segmente Wellel und Welle2 verglichen. Dies wird fortgesetzt, bis alle wahrscheinlichen Segmente verglichen sind. Wie bereits gesagt wurde, müssen nur Wellenlängen innerhalb einer bestimmten Bandbreite berücksichtigt werden. Segmente werden zuerst getestet, ob sie wahrscheinliche Übereinstimmungen sind, bevor sie zum Vergleich an das Kästchen **63** gesendet werden. Die Spitzen Start und Aufteilen müssen ebenfalls starke Spitzen sein.

[0089] Unter erneuter Bezugnahme auf Fig. 5 ist t_{17} aktuell, und die in der nachstehenden Tabelle A gezeigten Wellensegmente werden verarbeitet.

Tabelle A. Wellensegmente

Welle1 Welle(start, aufteilen)	Welle2 Welle(start, aktuell)	Welle1 Welle(start, aufteilen)	Welle2 Welle(start, aktuell)
Welle(15, 16)	Welle(16, 17)	Welle(13, 14)	Welle(14, 17)
Welle(14, 16)	Welle(16, 17)	Welle(12, 14)	Welle(14, 17)
...	...	...	...
Welle(8, 16)	Welle(16, 17)	Welle(6, 14)	Welle(14, 17)
...	...	...	...
Welle(14, 15)	Welle(15, 17)	Welle(8, 9)	Welle(9, 17)
Welle(13, 15)	Welle(15, 17)	Welle(7, 9)	Welle(9, 17)
...	...	...	...
Welle(7, 15)	Welle(15, 17)	Welle(1, 9)	Welle(9, 17)

[0090] Der Pseudocode für das Kästchen **62** ist in Fig. 8 angegeben. In den Zeilen 1 – 4 lauten Wellel und Welle2 gemäß der obigen Definition: die Länge()-Funktion ist so definiert, dass sie die Wellenlänge, wie sie oben angegeben wurde, ist; und die Abtastwert()-Funktion gibt den Wert des Eingangsabtastwerts bei der ge-

gebenen Spitze zurück. Die Zeile 6 initialisiert ein Flag, das aufzeichnet, ob eine Übereinstimmung vorlag. Zeilen 7 bis 12 iterieren durch alle Wellensegmente, die in Frage kommende Kandidaten für eine Übereinstimmung sind. Die Wellen müssen innerhalb der Wellenlängen liegen, die die menschliche Stimme erzeugen kann (Zeilen 8, 9). Sie müssen ungefähr dieselbe Länge aufweisen (Zeile 10). Sie müssen ungefähr bei derselben Höhe auf dem Graphen beginnen und enden (Zeilen 11 und 12). Falls alle diese Kriterien erfüllt sind, werden die Wellen anschließend verglichen (Zeile 13). Die Vergleichsprozedur wird in dem folgenden Absatz ausführlich beschrieben, kurz gesagt dehnt sie jedoch die Wellen auf dieselbe Länge aus und subtrahiert anschließend eine von der anderen. Die sich aus dieser Subtraktion ergebende Differenz wird verwendet, um ihre Ähnlichkeit zu bewerten: je näher die Differenz bei Null liegt, desto ähnlicher sind die zwei Wellen. Wenn die zwei Wellen ähnlich genug sind (Zeilen 14 und 15), dann werden sie als passend bzw. übereinstimmend betrachtet. Das Kriterium dafür, ob sie zusammenpassen oder nicht, hängt davon ob, ob derzeit ein Ton gespielt wird. Wenn kein Ton gespielt wird, wird ein strengerer Standard angewendet, was gewährleistet, dass das Spielen bei der richtigen Frequenz beginnt. Nachdem ein Ton zu spielen begonnen wurde und die ungefähre Frequenz festgelegt wurde, wird ein lockererer Standard angewendet. Diese zwei verschiedenen Standards sind aufgrund des Frequenzberechnungs- und Korrekturmoduls (FCCR – Frequency Calculation and Correction Module) nötig: Wie nachfolgend ausführlicher erläutert wird, zwingt das FCCR, nachdem ein Ton festgelegt wurde, jede neue Frequenz, nahe bei der vorherigen zu liegen. Somit ist es viel wichtiger, bei Beginn des Tons die Frequenz richtig hinzubekommen.

[0091] Wenn die Segmente zusammenpassen, wird anschließend die Grundwellenlänge, die sie darstellen (der Durchschnitt ihrer Einzellängen) berechnet (Zeile 16). Wenn diese Wellenlänge ungefähr das Doppelte der besten bisher abgestimmten Wellenlänge beträgt (Zeile 17), so ging die Suche zu weit, und Wellel und Welle2 bestehen jeweils aus zwei vollständigen Grundwellenformen. In diesem Fall endet die Verarbeitung, und die neue Frequenz wird zurückgegeben (Zeile 18). Andernfalls wird die Differenz zwischen den Segmenten mit vorherigen Differenzwerten verglichen (Zeile 19). Falls sie die bisher niedrigste ist, wird sie behalten (Zeilen 20 und 21), und das Übereinstimmung-Flag wird auf WAHR gesetzt.

[0092] Obwohl sich das hier beschriebene bevorzugte Frequenzerfassungsverfahren auf die Identifizierung von Spitzen stützt, könnte es sich ebenso gut auf die Identifizierung eines beliebigen anderen Unterscheidungsmerkmals, z. B. Nulldurchgänge, stützen. Im Fall von Nulldurchgängen ist das Analog einer starken Spitze (im Kästen **61**) ein Nulldurchgang mit einer großen positiven oder negativen Neigung.

Vergleichen zweier Wellensegmente

[0093] Das Kästchen **63**, das zwei Segmente vergleicht, nimmt zwei Wellensegmente, dehnt das zweite aus oder verkleinert es, so dass es dieselbe Länge wie das erste aufweist, und addiert ihre Abtastwertunterschiede auf. Statt die Unterschiede über jeden Abtastwert in beiden Wellen zu summieren, wird lediglich eine kleine Anzahl gleichmäßig verteilter Abtastwerte (als Prüfpunkte) gewählt, was die Verarbeitung beschleunigt. Der Abstand zwischen jedem Prüfpunkt beträgt ungefähr N_SAMPLES_PER_CHECKPOINT (N_ABTASTWERTE_PRO_PRÜFPUNKT). Der Pseudocode für das Kästchen **63** ist in **Fig. 9** gezeigt. Die zwei Wellensegmente werden als Wellel und Welle2 bezeichnet. Zeile 2 berechnet die Anzahl der Prüfpunkte auf der Grundlage der Länge der Wellel. Das Bodensymbol „[]" bedeutet, auf die nächstliegende Ganzzahl abzurunden. Der Wert von wavelength ratio (Wellenlängen Verhältnis) stellt die Länge der Welle2 im Vergleich zur Länge der Wellel dar (Zeile 3). Die während-Schleife in den Zeilen 5, 6 und 13 iteriert durch alle Prüfpunkte. Bei Zeile 7 ist t_1 die Zeit des i-ten Prüfpunkts in Wellel. t_2 ist die Zeit des i-ten Prüfpunkts in Welle2 – die auf t_1 beruht, jedoch erweitert oder verringert ist, um derselben Position in Welle2 zu entsprechen. Zeilen 9 und 10 finden die Abtastwerte für Wellel und Welle2 an dem i-ten Prüfpunkts. In Zeile 11 wird die Differenz zwischen den zwei Wellen mit dem Betrag ihrer Differenz an diesem Prüfpunkt aktualisiert. In den Zeilen 12–14 wird der durchschnittliche Betrag der zwei Abtastwerte berechnet, und das Maximum dieses Werts wird für die gesamte Welle beibehalten. Am Ende, in Zeile 16, wird die Summe der Differenzen der Welle sowohl für Länge als auch Höhe normiert, so dass der Effekt der Prozedur sowohl für hohe als auch für niedrige Frequenzen und sowohl für laute als auch für leise Signale derselbe ist.

Die Frequenzberechnungs- und -korrekturroutine (FCCR)

[0094] Während des Verlaufs eines Tons ändert sich die Frequenz zwischen aufeinander folgenden Spitzen niemals drastisch. Die menschliche Stimme kann die Frequenz über derartige geringe Zeitskalen einfach nicht sehr stark ändern, und diese Tatsache kann genutzt werden, um ein gewisses Maß an Frequenzkorrektur zu liefern. Wenn sich die aktuelle Frequenz beträchtlich von der zuvor erfassten Frequenz unterscheidet, dann ist entweder die Erstere oder die Letztere falsch. Obwohl man auf viele verschiedene Arten mit dieser Situation

umgehen könnte, geht die Frequenzberechnungs- und -korrekturroutine (FCCR) (Kästchen **64** in Fig. 6) davon aus, dass die zuletzt erfasste Frequenz falsch ist, und ersetzt sie durch die zuvor erfasste Frequenz. (Besonders aus diesem Grund ist die Frequenzerfassung strenger, wenn ein Ton festgelegt wird, als nachdem der Ton bereits begonnen hat, siehe Fig. 8, Zeilen 14 und 15). Dadurch, dass in dem Eingang des HumHorns lediglich kleine Frequenzänderungen akzeptiert werden, scheint sich die Tonhöhe im Ausgang des HumHorns glatt und kontinuierlich zu verändern.

[0095] Der Pseudocode für die FCCR ist in Fig. 10 gezeigt. Die Zeile 2 berechnet die seit der letzten Wellenübereinstimmung verstrichene Zeit. Die Zeile 3 berechnet, welche Frequenz vorliegen wird, wenn die beste Wellenlänge akzeptiert wird, gemäß Gleichung (1). Die Zeilen 4 – 7 berechnen die prozentuale Frequenzdifferenz zwischen der zuletzt akzeptierten Frequenz und der neu vorgeschlagenen. Der Zähler ist der Größere der beiden, und zur Normierung wird 1 von dem Quotienten subtrahiert. Wenn im Kästchen **62** keine Übereinstimmung gefunden wurde, wird die Frequenz unverändert belassen (Zeile 9). Andernfalls wird ihre Zeit aufgezeichnet (Zeile 9), um bei einer späteren Iteration bei Zeile 2 wiederverwendet zu werden. Wenn die Frequenzänderung innerhalb der Geschwindigkeit lag, die die menschliche Stimme erreichen kann, so wird die Frequenz auf den neuen Wert geändert, andernfalls wird sie unverändert belassen.

[0096] Ein alternativer Lösungsansatz bezüglich der Frequenzkorrektur besteht darin, während eines Tons lediglich eine bedeutende Frequenzänderung zu erlauben, jedoch nur dann, wenn die Übereinstimmungsbewertung für die neue Frequenz sehr gut war, d. h. wenn die Differenz in Fig. 8, Zeile 13, sehr gering ist. Dies korrigiert einen durch den Sänger schlecht gestarteten Ton oder einen durch das HumHorn nicht korrekt erfassten Toneinsatz.

[0097] Die meisten Fehler bei der Frequenzerfassung sind um eine Oktave versetzt. Fig. 11 zeigt zwei gefilterte Signalverläufe, die von derselben Stimme aufgezeichnet wurden. Diese Signalverläufe sind besonders interessant, weil sie so sehr mehrdeutig sind und eine Herausforderung für den Frequenzdetektor sind. Die obere Welle weist eine Wellenlänge von knapp unter acht Millisekunden auf, aufgrund einer Formreplikation könnte sie jedoch ohne weiteres auch so interpretiert werden, als sei ihre Wellenlänge doppelt so hoch. Für die untere Welle gilt das Gegenteil. Sie weist eine Wellenlänge von knapp über sieben Millisekunden auf, könnte jedoch ohne weiteres auch so interpretiert werden, dass sie nur die Hälfte dieser Wellenlänge aufweist. Damit das FDM beide Wellenlängen korrekt erkennt, müssen die Parameter sorgfältig abgestimmt sein. Der vollständige Satz von Parameterwerten wird nachfolgend erörtert.

[0098] Ein weiteres Verfahren zum Korrigieren der Frequenz beinhaltet ein Verändern des Kästchens **62** in Fig. 6. Wenn die beste Übereinstimmung zu weit von der vorherigen Wellenlänge entfernt ist, könnte das Kästchen **62** die Übereinstimmung, die am nächsten bei der vorherigen Wellenlänge lag, zurückgeben. Es ist möglich, dass diese Wellenlänge, obwohl sie nicht die beste Übereinstimmung ist, die echte Grundwellenlänge war.

[0099] Die Frequenzkorrektur ist ein wichtiger Teil des Frequenzerfassungsprozesses, und es gibt eine Vielzahl verschiedener Techniken, um sie zu bewerkstelligen. Bei der Frequenzkorrektur hat der allgemeine Frequenzerfassungsalgorithmus die Anzahl von in Frage kommenden Frequenzen aus dem gesamten Spektrum, das die menschliche Stimme erzeugen kann, auf eine geringe Anzahl von Alternativen, üblicherweise zwei oder drei, reduziert. In diesen Fällen kann die Mehrdeutigkeit, die für den Mehrzweck-Frequenzerfassungsalgorithmus schwierig zu unterscheiden sein kann, für einen zweckgebundenen Algorithmus leichter sein. Zwei Verfahren, die für diese gelegentlichen Frequenzmehrdeutigkeiten verwendet werden können, sind insbesondere: (a) Prädikationsfiltern und andere parametrische Frequenzschätzungstechniken und (b) kontext-sensible probabilistische Verfahren. Im Fall von (a) erfordern diese Verfahren viele Vorab-Informationen über die Welle und sind somit für den Fall einer allgemeinen Frequenzerfassung schwer zu handhaben, sie sind jedoch präzise und effizient, wenn die in Frage kommenden Frequenzen auf eine geringe Anzahl reduziert werden können, wie bei dem Frequenzkorrekturszenario. Im Fall (b) kann der Kontext, d. h. können die Tonhöhen, die der Benutzer kürzlich gesungen hat, helfen, die Wahrscheinlichkeit, dass der Sänger versuchen wird, andere Tonhöhen zu singen, vorauszusagen. Diese Wahrscheinlichkeiten können zusammen mit der geringen Anzahl von Kandidaten, die durch den Frequenzerfassungsalgorithmus gefunden wurden, verwendet werden, um die Tonhöhe, die am wahrscheinlichsten versucht wird, auszuwählen. Wenn der Sänger bei den letzten paar Tönen beispielsweise in aufsteigenden Halbtönen nach oben gesungen hat, dann ist – wenn zwei mehrdeutige Alternativen, eine einen Halbton höher als die letzte Tonhöhe und die andere eine Oktave plus einen Halbton höher, gegeben sind – die Wahrscheinlichkeit größer, dass der Sänger eher die Erstgenannte und nicht die Letztgenannte beabsichtigte. Allgemein können Vorab-Informationen über die menschliche Stimme und die Wellenmuster, die sie erzeugen kann oder die sie wahrscheinlich erzeugen wird, dazu verwendet werden, die endgültige Entscheidung darüber zu treffen, welche Frequenz erfasst wurde.

Schätze Frequenzzuverlässigkeit

[0100] Kästchen **66**, Schätze Frequenzzuverlässigkeit, ist eine wichtige Routine, sie wird jedoch besser nachfolgend beschrieben, wenn der Kontext für ihre Verwendung klarer wird. Für den Moment sei angemerkt, dass das Kästchen **66** noch nicht beschrieben wurde, dass das Kästchen **66** jedoch in der nachstehenden Erläuterung, wenn eine Messung der Frequenzzuverlässigkeit benötigt wird, beschrieben und erklärt werden wird.

Bemerkungen zum FDM

[0101] Die gesamte Frequenzerfassungsprozedur ist rechentechnisch nicht sehr aufwendig – sie besteht hauptsächlich aus Summierungen über einen kleinen Teilsatz von in jüngerer Zeit erfolgten Zeitschritten. Dennoch ist dieser Lösungsansatz sehr effektiv darin, die richtige Grundfrequenz zu finden. Die meisten anderen Verfahren zur Frequenzerfassung stützen sich auf ein viel umfangreicheres Filtern des Signals. Spektralverfahren erfordern mehrere Größenordnungen mehr an Berechnung. Das FDM reagiert außerdem viel schneller als Spektralverfahren; es erfordert weit weniger Abtastwerte, bevor die Grundwellenlänge erfasst wird. Das FDM-Verfahren ist auf eine standardmäßige Autokorrelation bezogen, ist jedoch rechentechnisch weniger aufwendig. Während Autokorrelationsverfahren ein Punktprodukt verwenden, um eine Wellenformähnlichkeit zu messen, was eine große Zahl von Multiplikationen beinhaltet, verwendet das FDM die Summe von Differenzen. Das FDM spart auch beträchtlichen Rechenaufwand ein, indem es lediglich Wellensegmente vergleicht, die durch Wellenformmerkmale – z. B. Spitzen oder Nulldurchgänge – begrenzt werden.

Das Spiel- und Einsatzentscheidungsmodul (PADM)

[0102] Der Klang mancher Instrumente am Anfang eines Tons kann ziemlich bezeichnend sein. Beispielsweise weisen Blechinstrumente üblicherweise einen stechenden, rauchigen Klang auf. Dieser Teil des Tons wird als sein Einsatz bezeichnet. Die Frequenz, bei der der Einsatz eines Tons erfolgt, wird als seine Einsatzfrequenz bezeichnet. Insgesamt spielt das HumHorn in vier verschiedenen Fällen einen Einsatz:

- wenn ein abrupter Anstieg der Lautheit erfolgte, der von einer guten Frequenzerfassung begleitet war oder auf den eine solche bald folgte;
- wenn derzeit kein Ton gespielt wird, jedoch über eine gewisse Mindestzeit hinweg eine gute Frequenz erfasst wurde;
- wenn die Tonhöheerfassung kontinuierlich ist und die Tonhöhe sich über den Bereich hinausbewegt hat, in dem der Synthesizer auf der Grundlage der Einsatztonhöhe glatte Tonhöheänderungen erzeugen kann; und
- wenn die Tonhöhenachverfolgung diskret ist und die Tonhöhe sich zu weit von der Einsatztonhöhe entfernt hat und einer anderen Tonhöhe in dem ausgewählten Musikmodus zu nahe gekommen ist.

[0103] Der dritte und der vierte dieser Fälle beinhalten Leistungsfähigkeits- und Synthesizeroptionen. Diese Fälle werden durch das SSM (das Klangsynthetisierungsmodul) gehandhabt und werden nachfolgend ausführlich beschrieben. Die ersten zwei Fälle erfordern die Erfassung eines seitens des Spielers beabsichtigten Einsatzes. Diese Fälle werden durch das Spiel- und Einsatzerfassungsmodul (PADM) gehandhabt.

[0104] Wenn der Spieler summt, möchte er, dass das HumHorn zu bestimmten Zeiten einen Einsatz erzeugt. Der Spieler benutzt automatisch seine Zunge und Lippen, wobei er Konsonanten erzeugt, üblicherweise ‚d', ‚t', ‚l', ‚p', ‚b' und/oder ‚m', um einen Ton von dem nächsten zu trennen. Allgemein singt man Silben wie ‚dum, ba dum, badumpadumpadum' oder ‚doodle oo, doodle oo, doodle oo doo doo', um Töne voneinander abzugrenzen. Dadurch signalisiert der Spieler dem HumHorn intuitiv, einen Einsatz zu erzeugen. Das PADM kann diese Signale erfassen und auf dieselben ansprechen, indem es Einsätze an das SSM ausgibt.

[0105] In beiden oben genannten Fällen (1) und (2) sollte kein Einsatz ausgegeben werden, wenn ein Zweifel bezüglich der Frequenz des Signals besteht. Ferner muss der Spieler im Fall (2) seit einem gewissen sinnvollen Zeitraum eine gute Frequenz singen. Wie oben erwähnt wurde, wird somit eine Vorgehensweise zum Bestimmen der Zuverlässigkeit der erfassten Frequenz benötigt. Die Frequenzzuverlässigkeitsinformationen werden von dem Kästchen **66** der Fig. 6 hereingeleitet. Diese Vorgehensweise gibt einen von vier möglichen Werten zurück:

- SCHLECHT: Die aktuelle Frequenz ist unbekannt, und in letzter Zeit wurde keine zuverlässige Frequenz erfasst.
- UNGEWISS: In letzter Zeit wurde eine Frequenz erfasst, jedoch nicht an der jüngsten Spitze.
- GUT: An der jüngsten Spitze wurde eine Frequenz erfasst, aber in jüngerer Zeit bestanden bzw. bestehen Ungewissheiten.

• STABIL: In jüngerer Zeit wurde und wird konsequent eine Frequenz erfasst.

[0106] Die Routine zum Berechnen der Frequenzzuverlässigkeit wird nachfolgend beschrieben. Als erstes wird jedoch das restliche PADM beschrieben. Ein Diagramm der Beziehungen zwischen dem PADM und seiner verwandten Routinen ist in Fig. 13 angegeben. Neben der Frequenzzuverlässigkeit erfordert das PADM **46** ferner Informationen über die aktuelle Lautheit der Stimme des Spielers. Die Lautheit wird durch das LTM **47**, das Lautheitnachverfolgungsmodul, das nachfolgend beschrieben wird, berechnet. Das PADM benötigt ferner Informationen über kürzliche Änderungen der Lautheit, insbesondere ob die Lautheit in der jüngeren Vergangenheit plötzlich abrupt angestiegen ist. Diese Informationen kommen von der Kürzlicher-Lautheitssprung-Routine **130**, die nachfolgend beschrieben wird.

[0107] Der Pseudocode für das PADM ist in Fig. 14 angegeben. Das PADM gibt in zwei verschiedenen Fällen, entsprechend Fall (1) und Fall (2) oben, einen Einsatz aus. Zeilen 3 – 5 entsprechen Fall (1), Zeilen 7 – 10 entsprechen Fall (2). In dem ersten Fall wird ein Einsatz ausgegeben, wenn: (Zeile 3) kürzlich zumindest eine Frequenzübereinstimmung vorlag (die Frequenzstabilität GUT oder STABIL ist); (Zeile 4) ein abrupter Anstieg der Lautheit erfolgt ist; und (Zeile 5) seit dem letzten Einsatz ausreichend Zeit verstrichen ist. In dem zweiten Fall ist die Frequenzstabilität seit einer gewissen Zeit stetig (Zeile 7), das Signal ist laut genug (Zeile 8), derzeit wird kein Ton gespielt (Zeile 9) und genügend Zeit ist verstrichen, seit der Ton abgeschaltet wurde (Zeile 10). In beiden diesen Fällen werden Flags gesetzt, um anzugeben, dass nun ein Ton spielen sollte (Zeile 12) und dass ein Einsatz ausgegeben werden sollte (Zeile 13). Die Zeit wird zur Verwendung bei der nächsten Iteration bei Zeile 5 aufgezeichnet (Zeile 15). Wenn weder Fall 1 noch Fall 2 zutrifft, wird kein Einsatz ausgegeben (Zeile 17). Wenn für die letzte Spitze ebenfalls keine Übereinstimmung vorlag oder die Lautheit auf einen ausreichend niedrigen Pegel abgesunken ist (Zeile 18), dann wird das Spielen abgeschaltet (Zeile 19), und die Zeit wird zur Verwendung bei der nächsten Iteration bei Zeile 10 aufgezeichnet (Zeile 20).

Schätzung der Frequenzzuverlässigkeit

[0108] Die vorliegende Erörterung betrifft Kästchen **66** in Fig. 6, Schätze Frequenzzuverlässigkeit. Es mag hilfreich sein, zu Referenzzwecken zu Fig. 6 und dem dazugehörigen Text zurückzugehen. Jede Spitze wird ausgewertet, wenn sie auftritt, d. h. immer dann, wenn sich ein Abtastwert bei einer STARKEN oder einer SCHWACHEN Spitze befindet. Schwache Spitzen sind relativ häufig. Obwohl sie nicht verwendet werden, um eine Formübereinstimmung und Frequenzerfassung einzuleiten, sind sie dennoch ein hilfreicher Teil des Formabstimmungsvorgangs, da sie wesentliche Informationen über die Form der Welle liefern. Wenn starke, übereinstimmende Spitzen, d. h. Spitzen, für die das FDM eine Übereinstimmung fand, durch eine geringe Anzahl schwacher Spitzen getrennt sind, dann ist dies ein gutes Zeichen und weist auf ein verlängertes, stabiles Signal bei der erfassten Frequenz hin. Wenn jedoch zu viele schwache Spitzen hintereinander vorliegen, dann ist dies ein schlechtes Zeichen. Es gibt an, dass das Signal selbst schwach ist oder verloren gegangen ist. Ein schwaches oder verloren gegangenes Signal ist ein normaler Teil der Erfassung. Es passiert am häufigsten, wenn der Spieler aufhört, einen Ton zu singen, oder zwei Töne trennt, indem er einen Konsonanten ausspricht, was für ein zusätzliches Rauschen in dem Signal sorgt. Ein weiteres schlechtes Zeichen, d. h. ein Hinweis darauf, dass die Frequenz verloren gegangen ist, liegt vor, wenn eine starke Spitze gefunden wird, das FDM jedoch keine Formübereinstimmung für dieselbe finden kann. Somit kann die Zuverlässigkeit des Signals gemäß guten Zeichen und schlechten Zeichen eingestuft werden. Die guten Zeichen sind starke, übereinstimmende Spitzen. Die schlechten Zeichen sind nicht-übereinstimmende Spitzen und Abfolgen zu vieler schwacher Spitzen hintereinander. Die Frequenz ist STABIL, wenn hintereinander zumindest drei gute Zeichen und keine schlechten Zeichen vorliegen. Sie ist SCHLECHT, wenn hintereinander zumindest fünf schlechte Zeichen und keine guten Zeichen vorliegen. Wenn sie weder SCHLECHT noch STABIL ist, die aktuelle Spitze jedoch eine starke, übereinstimmende Spitze ist, so ist die Frequenzzuverlässigkeit GUT. Wenn keiner dieser Fälle zutrifft, dann ist die Zuverlässigkeit UNGEWISS.

[0109] Fig. 15 zeigt den Pseudocode zum Schätzen der Frequenzzuverlässigkeit. Zeilen 2 – 15 zählen die guten Zeichen und schlechten Zeichen. Zeilen 16 – 23 klassifizieren sie in eine Zuverlässigkeitsschätzung. Wenn die aktuelle Spitze schwach ist (Zeile 2), so wird die Anzahl aufeinander folgender schwacher Spitzen inkrementiert (Zeile 3). Wenn die Anzahl aufeinander folgender schwacher Spitzen zu hoch ist (ein schlechtes Zeichen), dann sollte der Schlechte-Zeichen-Zähler inkrementiert werden (Zeile 5), und der Gute-Zeichen-Zähler sollte auf Null zurückgesetzt werden (Zeile 6). Ferner sollte das Zählen aufeinander folgender schwacher Spitzen erneut beginnen (Zeile 7). Andernfalls muss die Spitze stark sein, da sie nicht schwach ist (Zeile 8). Wenn keine Übereinstimmung gefunden wurde (schlechtes Zeichen), dann wird der Schlechte-Zeichen-Zähler erneut inkrementiert (Zeile 9), der Gute-Zeichen-Zähler wird zurückgesetzt (Zeile 10) und der Schwache-Spitzen-Zähler wird zurückgesetzt (Zeile 11). Wenn jedoch doch eine Übereinstimmung gefunden wurde (gutes

Zeichen), so wird der Gute-Zeichen-Zähler inkrementiert (Zeile 13), der Schlechte-Zeichen-Zähler wird zurückgesetzt (Zeile 14), und der Schwache-Spitzen-Zähler wird wiederum zurückgesetzt (Zeile 15). Bei Zeile 16 beginnt die Klassifizierung. Wenn auf dem Weg zur aktuellen Spitze fünf oder mehr schlechte Zeichen hintereinander vorliegen, so ist die Frequenzzuverlässigkeit SCHLECHT (Zeile 17). Wenn drei oder mehr gute Zeichen vorliegen, so ist die Zuverlässigkeit STABIL (Zeile 19). Wenn sie weder SCHLECHT noch STABIL ist, sondern die aktuelle Spitze eine starke, übereinstimmende Spitze ist, so ist die Zuverlässigkeit GUT (Zeile 21). Wenn keiner dieser Fälle zutrifft, dann ist die Zuverlässigkeit UNGEWISS (Zeile 23).

[0110] Die Anzahlen guter Zeichen (3) und schlechter Zeichen (5) sind deutlich willkürlich und eine Angelegenheit des Aufeinanderabstimmens. Die Kriterien für gute Zeichen und schlechte Zeichen könnten im Prinzip auch verbessert werden, um andere Indikatoren zu umfassen.

Erfassung von Lautheitssprüngen

[0111] Die letzte durch das PADM benötigte Komponente ist die Routine zum Erfassen eines kürzlichen Lautheitssprungs. Diese Routine gibt WAHR zurück, wenn kürzlich ein Lautheitssprung (ein plötzlicher Anstieg der Lautheit der Stimme des Spielers) erfolgte, der nicht bereits als Grundlage für einen Einsatz verwendet wurde. Ein Lautheitssprung gilt immer dann als erfolgt, wenn die aktuelle Lautheit wesentlich größer ist als jegliche vorherige Lautheit in der jüngeren Vergangenheit, d. h. wenn

$$L(t) > SURGE_RATIO \cdot \min(L(t - WINDOW_SIZE),..., L(t - \varepsilon), \quad (2)$$

wobei L(t) die aktuelle Lautheit ist, wie sie anhand eines der nachfolgend zu beschreibenden Verfahren berechnet wird, SURGE_RATIO das Verhältnis ist, um das die aktuelle Lautheit die frühere Lautheit übersteigen muss, WINDOW_SIZE die Länge der „jüngeren Vergangenheit" ist, d. h. die maximale Zeit, in der ein Lautheitssprung stattfindet, min() den minimalen Wert seiner Argumente zurückgibt und ε = 1/R die Zeit zwischen Abtastwerten ist, wobei R die Abtastrate ist. Wenn ein Lautheitssprung erfolgt, wird ein Flag gesetzt, und die Zeit wird aufgezeichnet. Die Routine gibt dann WAHR für Sprung_Zeit-Sekunden zurück, oder bis das Flag auf Null gesetzt wird (**Fig. 14**, Zeile 14).

[0112] Die „min"-Funktion in der Gleichung 0 ist im Vergleich zu dem Rest der Funktionen des HumHorns rechentechnisch aufwendig. Ein Verfahren, das diesen Prozess beschleunigen kann, besteht darin, die Lautheitswerte in Intervallbereiche aufzuteilen, von denen jeder eine Bandbreite von Werten darstellt. Wenn ein neuer Lautheitswert ankommt, wird der Intervallbereich, dem er entspricht, inkrementiert. WIN-DOW_SIZE (FENSTER_GRÖSSE) Sekunden später, wenn der Wert das Fenster verlassen sollte, wird der Intervallbereich dekrementiert. Der minimale Wert in dem Fenster liegt in der Bandbreite des niedrigsten, nicht Null betragenden Intervallbereichs. Alternativ dazu könnten die Intervallbereiche auf eine Liste der tatsächlichen durch diesen Intervallbereich dargestellten Werte zeigen. In der Tat könnten die Werte in einer beliebigen standardmäßigen, in eine Reihenfolge gebrachten Datenstruktur gespeichert werden, deren Speicherzeit O(log n) beträgt. Alternativ dazu kann für diesen Vergleich ein Teilsatz vergangener Lautheitswerte verwendet werden, z. B. diejenigen, die der Spitzenerfassung entsprechen.

Das Lautheitnachverfolgungsmodul (LTM)

[0113] Die unmittelbare und kontinuierliche Reaktion des HumHorns auf die von Moment zu Moment erfolgenden Anderungen der Lautheit der Stimme des Spielers liefert eine subtile, nuancierte Steuerung, die kein Tasteninstrument erreichen kann. Bei dem HumHorn ist die Steuerung der Lautstärke völlig intuitiv und natürlich; sie muss nicht erst vom Gehirn in die Finger übersetzt werden. Sie ist mühelos und automatisch. Eine reaktionsfähige Lautheitnachverfolgung ist außerdem sehr wichtig, damit das PADM schnelle sowie subtile Toneinsätze erfasst.

[0114] Eine Art und Weise, wie das SAM die Lautheit des Eingangssignals nachverfolgt, beinhaltet einen gleitenden Mittelwert des Betrags des Abtastwerts:

$$L(t) = (1 - K) \cdot |s(t)| + K \cdot L(t - \varepsilon), \quad (3)$$

wobei L(t) die Lautheit zur Zeit t ist, s(t) der Abtastwert zur Zeit t ist, || den Absolutwert angibt und $0 < K << 1$. L(t) ist einfach eine Bahn oder ein Tiefpassfilter des Betrags des Abtastwerts. Dieses Verfahren ist ausreichend, um langsame Änderungen der Lautheit nachzuverfolgen. Es ist jedoch nicht ausreichend, um rasche Änderungen nachzuverfolgen. Um eine rasche Abfolge von Toneinsätzen zu erfassen, ist es notwendig, rasche

Änderungen der Lautheit nachzuverfolgen.

[0115] Eine unkomplizierte Art und Weise, eine reaktionsfähigere Lautheitnachverfolgung zu bekommen, besteht einfach darin, alle Abtastwerte in der jüngeren Vergangenheit, d. h. über ein Fenster der jüngsten M Schritte, anzusehen. Die Lautheit wird einfach auf den Absolutwert des Abtastwerts in dieser Bandbreite eingestellt, dessen Betrag (d. h. Entfernung von der Mittellinie) am größten ist:

$$L(t) = \max(|s(t - M)|, \ldots, |s(t)|), \quad (4)$$

wobei M ungefähr die maximal erwartete Wellenlänge für alle Benutzer ist. Wenn also die Amplituden der Wellen größer werden, nimmt die Lautheit zu, und wenn sie kleiner werden, nimmt die Lautheit ab. Wenn kein Eingangssignal vorliegt, liegt die Lautheit nahe bei Null.

[0116] Die Gleichung 0 ist potentiell rechentechnisch aufwendig zu implementieren, obwohl manche Optimierungen entworfen werden können. Ein anderes Verfahren, das einen sehr geringen rechentechnischen Mehraufwand erfordert, verwendet einfach den tatsächlichen Mittelwert (nicht eine Bahn) der Abtastwertbeträge über die letzten M Schritte

$$\begin{aligned} L(t) &= \sum_{j=t-k}^{t} |s(j)| / M \\ &= L(t - \varepsilon) + \left[|s(t)| - |s(t - M)|\right] / M . \end{aligned} \quad (5)$$

[0117] Dieser Lösungsansatz ist rechentechnisch trivial, da er bei jedem Zeitschritt lediglich eine Addition und eine Subtraktion erfordert. Die Division kann ignoriert werden, da M immer gleich ist. Er beinhaltet ferner weniger Programmcode als die anderen Lösungsansätze, was je nach den Hardware-Beschränkungen eine wichtige Überlegung sein kann. Ein ähnliches, aber etwas anderes Verfahren misst die Küstenlinienlänge der Welle, um den Grad an Aktivität oder Energie, die sie aufweist, zu schätzen:

$$\begin{aligned} L(t) &= \sum_{j=t-k}^{t} |s(j) - s(j - \varepsilon)| / M \\ &= L(t - 1) + \left[|s(t) - s(t - \varepsilon)| - |s(t - M) - s(t - M - \varepsilon)|\right] / M . \end{aligned} \quad (6)$$

[0118] Ein ausgefeilteres Verfahren der Lautheitnachverfolgung bewerkstelligt das, was die Gleichung 0 beschreibt, ist jedoch rechentechnisch weit weniger intensiv. Auch dieses Verfahren findet den maximalen Wert in dem vorangehenden Zeitfenster (der Größe M), es aktualisiert jedoch seine Auswertung der Lautheit nur, wenn eine starke oder schwache Spitze ankommt (statt bei jedem Zeitschritt). Als weitere Optimierung sind die einzigen zur Auswertung verwendeten Abtastwerte diejenigen an erkannten Spitzen. Dieser Spitzenbasierte Lösungsansatz passt gut zu dem Spitzenbasierten FDM. Der in Fig. 28 gelieferte C++-Code implementiert dieses Verfahren.

[0119] Bei jedem der obigen Verfahren, vor allem aber bei dem letzten, können Frequenzinformationen verwendet werden, um M dynamisch anzupassen, d. h. die Größe des Abtastfensters. Wenn die Grundfrequenz hoch ist, können kürzere Fenster verwendet werden; wenn sie niedrig ist, können längere Fenster verwendet werden. Dadurch wird das Modul reaktionsfähiger in Bezug auf Änderungen der Lautheit. Wenn die richtige Frequenz immer bekannt wäre, dann ist M idealerweise einen Zeitschritt kleiner als die Grundwellenlänge. Dies ist darauf zurückzuführen, dass die volle Wellenlänge auch die maximale Spitze von der vorherigen Grundwelle enthalten könnte. Wenn die Lautheit in der Tat immer nur dann aktualisiert wird, wenn das FDM eine Übereinstimmung für eine Spitze findet, dann sollte M alles nach der Spitze umfassen, das übereinstimmte, d. h. alles nach der Aufteilen-Spitze der Fig. 8 für die am besten übereinstimmende Wellel und Welle2.

[0120] Wenn keine Frequenzinformationen zur Verfügung stehen, könnte die längste erwartete Wellenlänge für M verwendet werden. Dadurch wird eine übermäßige Ansprechempfindlichkeit vermieden, wo rasche Änderungen der Lautheit erfasst werden, die in dem Signal eigentlich nicht existieren, was zu häufige Toneinsätze durch das PADM bewirken könnte. Jedoch kann eine größere Ansprechempfindlichkeit durch eine Vielzahl von Heuristika erhalten werden, die die Wellenlänge ausgehend von teilweise oder völlig unzuverlässigen Frequenzinformationen zu schätzen suchen. Wenn für mehrere Spitzen hintereinander keine Übereinstimmung gefunden wurde, dann beginnt die richtige Größe für M unsicher zu werden. In diesem Fall besteht eine gute Heuristik darin, bei dem etwa 0,9-fachen der zuletzt erfassten Grundwellenlänge mit M zu beginnen. Wenn die Frequenz unzuverlässiger wird, kann M als Funktion g(Δt) der Zeit erhöht werden, die seit der letzten GUT-Frequenzzuverlässigkeitsmessung verstrichen ist. Das heißt

$$M(t) = \min[w_{max}, 0{,}9\ W(t_{lm})/g(t - t_{lm})] \quad (7)$$

wobei w_{max} die längste erwartete Wellenlänge ist, t die aktuelle Zeit ist, t_{lm} der letzte Zeitpunkt war, zu dem das FDM eine Übereinstimmung erfasste, und W(x) die Wellenlänge zum Zeitpunkt x ist. Dieses Verfahren ist besonders nützlich während der Lautheitssenken, die das Aussprechen von Konsonanten begleiten, wenn die Frequenzerfassung unzuverlässig werden kann.

[0121] Da eine Einsatzerfassung zum großen Teil auf einer Lautheitnachverfolgung beruht, sollte das Lautheitnachverfolgungsmodul im Idealfall darauf abgestimmt sein, wie empfindlich die Einsatzerfassung gemäß dem Wunsch des Spielers sein soll. Insbesondere sollte g(Δt) von den stimmlichen Fähigkeiten des einzelnen Spielers abhängen. Es gibt eine Grenze bezüglich dessen, wie schnell jemand die Tonhöhe seiner Stimme verändern kann. Man geht davon aus, dass die meisten Menschen die Tonhöhe nicht schneller als bei einer Rate von etwa einer Oktave in 40 Millisekunden verändern können. Wenn das Ziel darin besteht, falsche Einsätze zu minimieren, so kann g() den Schlimmster-Fall-Wert berechnen, so als ob die Frequenz bei dieser maximalen Rate in der Tat abfallen würde, wobei in dem Moment der letzten Übereinstimmung des FDM begonnen wird. Um jedoch den Genuss für einen großen Markt von Benutzern zu erhöhen, ist es vorzuziehen, M langsamer als bei dieser maximalen Rate zu verändern. Dies ermöglicht eine ansprechempfindlichere Einsatzerfassung zu dem Preis, dass bei denjenigen Benutzern, die ihre Tonhöhe während Konsonantenklängen gerne sehr rasch ändern, möglicherweise einige zusätzliche Einsätze erzeugt werden. Es ist zweckmäßig, beim Umgang mit Frequenzen und Wellenlängen aufgrund der logarithmischen Wahrnehmung von Tonhöhen durch das menschliche Ohr exponentielle Funktionen zu verwenden, und somit ist es zweckmäßig, folgendes zu definieren:

$$g(\Delta t) = 2^{\Delta t / \Delta t_d} \quad (8)$$

wobei Δt_d die Zeit ist, die benötigt wird, bis sich die Wellenlänge verdoppelt, d. h. bis die Frequenz eine Oktave abnimmt. Der Parameter Δt_d kann angepasst werden, um die angenommene Geschwindigkeit der Veränderung der Tonhöhe des Spielers widerzuspiegeln. In dem oben erwähnten Schlimmster-Fall-Szenario ist Δt_d 0,04 Sekunden, und g(0,040) = 2. Natürlich kann die Wachstumsrate des Abtastfensters neben den Gleichungen (7) und (8) auf verschiedene Arten ausgedrückt werden, ohne von der essentiellen Wesensart des Lösungsansatzes abzuweichen.

[0122] Fig. 12 zeigt ein typisches Profil von Lautheitswerten, die unter Verwendung der Gleichungen (7) und (8) erhalten werden. Das Lautheitsprofil **122** ist der entsprechenden Sequenz von Abtastwertbeträgen **123** überlagert. Man beachte, dass die Lautheit nur an den Spitzen aktualisiert ist, wie bei dem obigen Verfahren beschrieben wurde.

Weitere technische Einzelheiten

[0123] Im Einklang mit dem Pseudocode-Konzept wurden bei dem oben dargestellten Code bestimmte technische Einzelheiten weggelassen. Eine derartige Einzelheit ist die Verwendung von Ringpuffern, um die neuen Daten zu enthalten. Es besteht natürlich kein Erfordernis, alle zuvor aufgezeichneten Daten zu behalten. Alte Daten werden einfach ringförmig mit neuen Daten überschrieben. Diese Technik gilt für alle Informationen, die über eine Reihe von Iterationen angesammelt werden. Die Größe des Ringpuffers ist groß genug ausgelegt, um alle Informationen zu enthalten, die notwendig sind, die längstmögliche Wellenlänge, die die menschliche Stimme hervorbringen kann, w_{max}, zu verarbeiten. Allgemein müssen die Puffer genügend Daten halten, um einen Zeitraum von knapp über 2 w_{max} abzudecken.

[0124] Um mögliche Mehrdeutigkeitsprobleme, die sich aus der Unvollkommenheit des Pseudocodes ergeben können, zu vermeiden, ist in den Fig. 19 – Fig. 32 ein funktionierender C++-Programmcode enthalten.

Das Klangsynthetisierungsmodul

[0125] Wie in Fig. 3 gezeigt wurde, empfängt der Klangsynthesizer die folgenden Eingänge von dem SAM: Ton ein/aus, Frequenz, Lautheit und Einsatz. Von dem mit den Fingern betätigten Steuersystem (FAC-System) empfängt er Parameter von dem Benutzer, die Präferenzen bezüglich Instrument, Oktave/Versatz, diskret gegenüber kontinuierlich und Musikmodus sowie vielleicht anderer, hier nicht erwähnter Steuerungen festlegen. Diese Eingänge und ihre Beziehungen werden nun ausführlich beschrieben. Der Ausgang des SSM ist ein Strom von Ausgangsabtastwerten, die zur Umwandlung in das Ausgangssignal an den DRC gesendet werden.

[0126] Die innere Struktur des SSM **38** ist in Fig. 16 angezeigt. Das SSM besteht aus zwei Hauptkomponen-

ten: dem Nachrichtenprozessor (MP) **160** und dem Klangerzeuger (SG) **161**. Die Kästchen Tonhöheumwandlung und Lautstärkeumwandlung sind untergeordnete Funktionen, die nachfolgend beschrieben werden. Der MP nimmt die durch das SAM und die FAC erzeugten Informationen und erzeugt Nachrichten, die er an den SG sendet. Das auffallendste an dem SSM ist die asynchrone Beziehung zwischen dem Nachrichtenprozessor und dem Klangerzeuger. Der MP empfängt in regelmäßigen Abständen, vorzugsweise 8.000 Hz, 11.025 Hz oder 22.050 Hz, Signale von dem SAM, und der SG erzeugt in regelmäßigen Abständen, vorzugsweise bei derselben Rate, Klangabtastwerte. Jedoch können die Nachrichten nicht in regelmäßigen Abständen von dem MP an den SG gesendet werden. Stattdessen werden Nachrichten nur dann gesendet, wenn der Ausgang aus dem SG geändert werden muss.

[0127] Der SG erzeugt Klänge von einem Instrument, und zwar immer jeweils einen Ton. Er kann selbst und ohne weitere Hilfe ein Ausgangssignal, d. h. eine Serie von Ausgangsabtastwerten, erzeugen, das das angeforderte Instrument imitiert, wie es den angeforderten Ton mit der angeforderten Lautstärke spielt. Wenn der Ton zu spielen beginnt, spielt er weiter, bis er abgeschaltet wird. Der MP sendet Nachrichten an den SG, die ihm sagen, dass er einen Ton beginnen oder beenden soll. Während ein Ton spielt, kann der MP Nachrichten senden, um die Tonhöhe und die Lautstärke des Tons zu ändern. Der MP kann auch Nachrichten senden, die dem SG mitteilen, welches Instrument imitiert werden soll.

[0128] Die Erörterung beschreibt die Tonhöhe- und Lautstärkeumwandlungsfunktionen, den Nachrichtenprozessor und den Klangerzeuger ausführlich.

Die Tonhöhe- und Lautstärkeumwandlungsfunktionen

[0129] Die Tonhöheumwandlungsfunktion **162** nimmt die durch das SAM erzeugte Frequenz und transformiert sie für den MP in eine Tonhöhe. Obwohl die Tonhöhe und die Frequenz oft dazu verwendet werden, dasselbe zu benennen, besteht ein subtiler Unterschied. Die Frequenz tritt natürlich auf; die Tonhöhe ist vom Menschen gemacht. Die Frequenz beschreibt Schall als physikalisches Phänomen (Zyklen pro Sekunde). Die Tonhöhe ist psychophysikalisch und beschreibt, wie wir Schall wahrnehmen. Wenn zwei Frequenzen eine Oktave auseinander liegen, weisen sie ein feststehendes Verhältnis, d. h. 2, auf. Im Gegensatz dazu ist die Tonhöhe die Position einer Hörfrequenz auf einer linearen Musiktonleiter, z. B. einem Notensystem oder einer Klaviertastatur, wo zwei Tonhöhen, die eine Oktave auseinander liegen, durch eine konstante Anzahl von Schritten, z. B. 12, getrennt sind. Die lineare Tonhöhenskala ist dem menschlichen Verständnis eingängiger als die exponentielle Skala von Frequenzen. In dem vorliegenden Dokument wird Tonhöhe als kontinuierlicher Wert verstanden, der an jedem beliebigen Punkt auf einer linearen Musiktonleiter liegen kann. Ein Ton liegt ebenfalls auf einer linearen Musiktonleiter, weist jedoch einen diskreten Ganzzahlwert auf. Wie zuvor angegeben wurde, ist die Frequenz die Inverse der Wellenlänge: F = 1/W. Wenn man den Basis-2-Logarithmus einer Frequenz nimmt, wird diese in einen Tonhöhenwert auf einer linearen Skala umgewandelt, wobei zwei Tonhöhen, die eine Oktave auseinander liegen, einen Unterschied von Eins aufweisen. Eine Multiplikation mit 12 ergibt die normale chromatische 12-Ton-Skala. Somit gilt

$$P = 12 \log_2 (F), \tag{9}$$

wobei P die sich ergebende Tonhöhe ist und F die durch das FDM gegebene Frequenz ist. Ein Hinzuaddieren von 12 zu P erhöht es um eine Oktave. Ein Subtrahieren von 12 verringert es um eine Oktave.

[0130] Die Lautstärkeumwandlungsfunktion **163** nimmt den Lautheitswert von dem SAM und wandelt ihn in einen Lautstärkesteuerungswert für den MP um. Die Lautstärke kann jegliche monotone Funktion der Lautheit sein, obwohl die bevorzugte Funktion lautet:

$$\text{Lautstärke} = A \cdot (\text{Lautheit} - B) \tag{10}$$

wobei A ein Konstante-Skala-Faktor ist und B eine konstante systematische Abweichung ist. Aufgrund der logarithmischen Eigenschaften des Ohres mag es wünschenswert sein, dass die Lautstärkeumwandlung eine exponentielle Funktion ist.

Der Nachrichtenprozessor (MP)

[0131] Der MP empfängt Informationen von dem SAM und von der FAC. Von dem SAM empfängt er vier Werte: Ton ein/aus, Einsatz, Tonhöhe und Lautstärke, wobei die letzten beiden umgewandelte Frequenz und Lautheit sind, wie soeben beschrieben wurde. Die Informationen von dem SAM kommen synchron an: 4 Werte bei

jedem Zyklus. Die FAC sendet Spielerpräferenzwerte, z. B. Instrumenten- und Oktaveneinstellungen. Die FAC-Informationen kommen auf asynchrone Weise an, immer dann, wenn der Benutzer einen oder mehr Parameter ändern möchte.

[0132] Beispielsweise könnte der Spieler eine Taste drücken, um das durch das SSM imitierte Instrument zu wechseln, oder um das SSM zu veranlassen, bei einem Versatz von einer oder mehr Oktaven von der gesungenen Tonhöhe zu spielen. Der MP speichert die jüngsten Einstellungen als interne Variablen und wendet sie immer dann entsprechend an, wenn Nachrichten an den SG gesendet werden. Bei Fig. 16 sind vier derartige Variablen gezeigt: Oktave **164**, kontinuierlich **165**, Modus **166** und neue Tonika **167**. Anforderungen von der FAC bezüglich eines Instrumentenwechsels erfordern keine beträchtliche Verarbeitung durch den MP und können bei der Ankunft gehandhabt werden. Sie werden einfach zu Nachrichten gebildet und direkt an den SG geleitet.

[0133] Die Variable „Oktave" hält einen Wert zwischen –3 und +3. Wenn sie nicht Null ist, gibt diese Variable an, dass das HumHorn eine Tonhöhe erzeugen sollte, die so und so viele Oktaven unter oder über der gesummten Tonhöhe liegt. Hier werden nur Oktavenversätze erörtert, im Prinzip könnte der Benutzer jedoch jeglichen Versatz von der gesungenen Tonhöhe festlegen, z. B. eine große Terz (4 Halbtöne) oder eine vollkommene Quinte (7 Halbtöne). Diese Nicht-Oktaven-Versätze können verwendet werden, um ansprechende und interessante parallele Melodien zur Begleitung der gesummten Tonhöhen zu erzeugen.

[0134] Die Variable „kontinuierlich", falls WAHR, gibt an, dass die durch das HumHorn gespielte Tonhöhe für diejenigen Menschen mit einer sehr guten Stimmkontrolle genau der gesummten Tonhöhe folgen sollte. Wenn die Stimme des Spielers ihre Tonhöhe um einen winzigen Grad verändert, sollte sich die ausgegebene Tonhöhe um denselben Grad ändern. Bei der kontinuierlichen Tonhöhenachverfolgung kann das Instrument allen subtilen Tonhöhevariationen, die durch den Spieler verwendet werden, einschließlich Vibrato, folgen. Folglich kann das Instrument manchmal eine unangenehme, weinerliche Qualität aufweisen, was bei manchen Instrumentenklängen ausgeprägter ist als bei anderen. Beispielsweise ist es fast unmöglich, ein Klavier wie ein Klavier klingen zu lassen, wenn man die kontinuierliche Tonhöhenachverfolgung durchführt. Es klingt letztlich eher wie ein Honky-Tonk-Klavier. Außerdem haben wenige Leute eine besonders gute Tonhöhenkontrolle über ihre Stimme. Aus diesen Gründen liefert das HumHorn die Option, das Instrument den Ton spielen zu lassen, der der Tonhöhe des Benutzers am nächsten kommt. Dann bleibt die Tonhöhe des Instruments stabil, auch wenn die Stimme des Spielers etwas wackelt. Wenn also kontinuierlich FALSCH ist, so sollte die gespielte Tonhöhe nach oben oder nach unten auf den nächstliegenden Ton in der Musiktonleiter oder in dem durch den Spieler ausgewählten Modus gerundet werden, wie nachfolgend beschrieben wird. Die Variablen „Modus" und „new_tonic" („neue_Tonika") werden ebenfalls nachfolgend beschrieben.

[0135] Ein Pseudocode für den Nachrichtenprozessor ist in Fig. 17 gezeigt. In Zeile 2 ist die Tonhöhe modifiziert, so dass sie die Tonhöheskala des SG sowie die aktuelle Oktavenvariable widerspiegelt. Man geht davon aus, dass der SG eine lineare Tonhöhenskala aufweist, die in Halbschrittintervalle diskretisiert ist, die den traditionellen Tönen auf einer Klaviatur entsprechen. Dies ist das durch das MIDI-Protokoll verwendete System. Der Anfangston in der Skala ist willkürlich und hängt von dem SG ab. Der Wert synthesizer_offset (Synthesizer_Versatz) ist die Differenz zwischen eine Tonhöhe auf der mathematisch abgeleiteten Tonhöhenskala, wie bei Gleichung 0 beschrieben ist, und der entsprechenden Tonhöhe für den SG. Dies ist für alle Tonhöhen ein konstanter Versatz. Für das MIDI-Protokoll entspricht die Frequenz 440 Hz dem 69. Ton auf der Klaviatur. In diesem Fall beträgt der Synthesizer-Versatz 12 $\log_2(440) – 69$ oder etwa 36,38 (etwas über drei Oktaven).

[0136] Bei Zeile 5 wird der der Tonhöhe am nächsten liegende Ton für den durch den Spieler gewählten Musikmodus berechnet. Ein Musikmodus bzw. eine Tonart ist ein Teilsatz der elf Halbtöne in einer Oktave. Beispiele sind: Dur, Moll, Blues und viele weitere esoterische Tonarten, z. B. dorisch, phrygisch, Ganzton- und fünfstufig. Der chromatische Modus besteht aus jedem Halbton in der Oktave, die mit 0 – 11 nummeriert sind. Die Durtonart besteht aus den folgenden Halbtönen: {0, 2, 4, 5, 7, 9, 11}. Der erste Ton in dem Modus (Ton Null) wird als Tonika bezeichnet, und alle Halbtöne in dem Modus sind ein Versatz von der Tonika. Die Modusvariable ermöglicht dem Benutzer, auszuwählen, welcher Modus zu verwenden ist. Indem er die der Variable „new_tonic" zugewiesene mit dem Finger betätigte Steuerung drückt, kann der Spieler die Tonika jeder beliebigen Tonhöhe zuweisen, die er derzeit singt. Wenn die Tonhöheverfolgung kontinuierlich ist, so wird der chromatische Modus verwendet (Zeilen 3 und 4), also wird der nächstliegende Halbton nachgeschlagen. Die nearest mode_note_routine (nächstliegender_Modus_Tonhöhe-Routine) wird nachfolgend beschrieben.

[0137] Die Zeilen 6 – 13 entscheiden, ob Gründe vorliegen, trotz der Tatsache, dass von dem SAM kein Ein-

satzsignal vorliegt, einen Einsatz auszugeben. Die zwei Fälle in den Zeilen 8 – 9 und den Zeilen 11 – 12 entsprechen den Fällen 3 bzw. 4, die oben erörtert wurden. Bei dem ersten dieser Fälle ist die Tonhöhenachverfolgung kontinuierlich (Zeile 8), und die Tonhöhe hat sich über den Bereich hinaus bewegt, in dem der Synthesizer glatte Tonhöheänderungen auf der Grundlage der Einsatztonhöhe erzeugen kann (Zeile 9). Die Einsatztonhöhe ist current_note (aktueller_Ton) bei Zeile 7, der bei einem vorherigen Einsatz eingestellt wurde (Zeile 16). Die Bandbreite, die die Tonhöhe überschritten hat, ist MAX_BEND_RANGE (MAX_BIEGUNG_BANDBREITE) in Zeile 9. In dem zweiten Fall (Zeilen 11 – 12) ist die Tonhöhenachverfolgung diskret, und die Tonhöhe liegt viel näher bei einem anderen Ton in dem Modus als bei dem Einsatzton (Zeile 12). Der Einsatzton für diesen Fall ist wiederum current_note. MAX_PITCH_ERROR (MAX_TONHÖHE_FEHLER), ein Wert zwischen 0,5 und 1,0, bestimmt, wie viel näher die Tonhöhe bei dem anderen Ton liegen muss. Ein Wert von 0,5 gibt an, dass die Tonhöhe auf den nächstliegenden Ton gerundet werden sollte. Ein Wert von mehr als 0,5 fungiert als eine Art Hysterese, wobei verhindert wird, dass sich der Ton verändert, wenn die Stimme des Spielers ein wenig instabil ist.

[0138] Die Zeilen 14 – 33 senden dem SG die entsprechende Nachricht für die aktuelle Situation, falls überhaupt. Wenn aus einem der oben angegebenen Gründe und somit für einen der vier oben beschriebenen Fälle ein Einsatz ausgegeben wurde, wird eine Nachricht gesendet, den neuen Ton bei der neuen Lautstärke zu spielen (Zeilen 14 – 22). Ob nun die Tonhöhenachverfolgung diskret oder kontinuierlich ist – der SG empfängt eine Nachricht, einen nearest_note (nächstliegender_Ton), einen Ganzzahltonwert, zu spielen. Wenn die Tonhöhenachverfolgung kontinuierlich ist, empfängt der SG ebenfalls eine Nachricht, die Tonhöhe um einen gewissen Betrag nach oben oder unten zu biegen, um mit der Eingangsfrequenz übereinzustimmen. Zeilen 15 und 16 speichern den Ton und die Lautstärke für zukünftige Bezugnahmen. Wenn der SG derzeit einen Ton spielt, sendet die Zeile 18 eine Nachricht an den SG, dass er aufhören soll. Die Zeile 19 gibt die Nachricht aus, den neuen Ton bei der neuen Lautstärke zu spielen. Wenn die Tonhöhenachverfolgung kontinuierlich ist (Zeile 20), wird der neue Ton dazu angepasst, zu der Tonhöhe der Stimme des Spielers zu passen (Zeile 21). Die Zeit des Einsatzes wird aufgezeichnet (Zeile 22).

[0139] Beginnend bei Zeile 24 liegt kein Einsatz vor, der Synthesizer spielt jedoch immer noch einen zuvor eingesetzten Ton. Wenn das SAM einen Ton-Aus-Befehl gesendet hat (Zeile 24), dann wird an den SG eine Nachricht gesandt, zu spielen aufzuhören (Zeile 25). Andernfalls spielt derzeit ein Ton und sollte weiterspielen, jedoch sollen vielleicht seine Tonhöhe oder Lautstärke angepasst werden (Zeilen 26 – 33). Da es eine Weile dauern kann, bis der SG seine Befehle verarbeitet, werden Veränderungen der Tonhöhe und der Lautstärke nur gelegentlich ausgesendet. Der Wert SG_REFRACTORY_PERIOD (SG_REFRAKTÄR_PERIODE) hängt von dem SG ab und legt fest, wie viel Zeit verstreichen muss, bevor eine neue Tonhöhe- oder Lautstärkenachricht gesendet werden kann. Wenn sich die neue Lautstärke beträchtlich von der bereits spielenden Lautstärke unterscheidet (Zeile 27), dann wird an den SG eine Nachricht gesendet, die Lautstärke an den neuen Wert anzupassen (Zeile 29), und die neue Lautstärke wird gespeichert (Zeile 28). Wenn die Tonhöhenachverfolgung kontinuierlich ist und sich die Tonhöhe geändert hat (Zeile 31), wird die Tonhöhe angepasst (Zeile 32). In beiden Fällen wird die Zeit zur erneuten Verwendung bei Zeile 26 aufgezeichnet (Zeilen 30 und 33).

[0140] Die adjust_pitch-Routine (Tonhöhe_Anpassen-Routine) hängt wiederum von dem SG ab. Für das MIDI-Protokoll ist es möglich, die Tonhöhe über eine gleitende Tonhöhenverschiebung anzupassen, sowie die maximal zulässige Bandbreite der gleitenden Tonhöhenverschiebung (MAX_BEND_RANGE (MAX_GLEITENDE_VERSCHIEBUNG_BANDBREITE)) anzupassen. Die adjust_pitch-Routine tut beides, falls erforderlich.

[0141] Die Funktion, die den am nächsten liegenden Moduston zurückgibt, ist in **Fig. 18** als Pseudocode gezeigt. Vier Modi sind am Anfang definiert, obwohl viele andere vorliegen könnten. Jeder Modus ist in Bezug auf die Halbtöne, aus denen er besteht, definiert, wobei bei der Tonika, bei der Position Null, begonnen wird und eine Oktave über der Tonika, bei Position 12, geendet wird. In der Durtonart liegt der zweite Ton zwei Halbtöne über der Tonika. Der nächste liegt zwei weitere Halbtöne über demselben, d. h. vier über der Tonika. Der nächste liegt einen weiteren Halbton darüber. Die Tonika selbst ist eine Ganzzahl zwischen 0 und 11 und ist ein Ton in der niedrigsten Oktave der linearen Tonhöhenskala. Der zwölfte Halbton über der Tonika ist eine Oktave über der Tonika, er hat jedoch denselben Platz in dem Modus wie die Tonika und wird ebenfalls als die Tonika betrachtet. In der Tat sind alle Modi oktavenblind, d. h. sie sind Versätze von der nächstliegenden Tonika darunter. Wenn also die Tonhöhe 38,3 ist und die Tonika 2 ist, so ist die nächstliegende Tonika unter 38,3 36 (2 + 12 + 12 + 12). Wenn die Variable new_tonic (neue_Tonika) eingestellt wird, so wird die Ganzzahl, d. h. der Halbton, die bzw. der der gegebenen Tonhöhe am nächsten ist, als Tonika gespeichert, jedoch auf die erste Oktave in der Skala reduziert, so dass sie bzw. er einen Wert zwischen 0 und 11 aufweist (Zeile 7). Die Variable „Versatz" ist die Differenz zwischen der Tonhöhe und der nächstliegenden Tonika unter derselben (Zeile 8). Bei Zeile 9

ist mode_note (Modus_Ton) (eine Ganzzahl) die Nummer in dem festgelegten Modus, die einem Versatz am nächsten liegt (eine Reelle). Die Differenz zwischen denselben (Zeile 10) ergibt bei Hinzufügung zu der ursprünglichen Tonhöhe den nächstliegenden Moduston (Zeile 11).

Der Klangerzeuger

[0142] Es gibt zwei Hauptarten, wie der Klangerzeuger implementiert werden kann: mit einem standardmäßigen MIDI-Modul (MIDI = Musical Instrument Digital Interface, digitale Musikinstrument-Schnittstelle) oder mit einem selbstentworfenen Synthesizer. Da die Anforderungen an dieses Modul weitaus geringer sind als die Fähigkeiten der meisten MIDI-Systeme, mag es vorzuziehen sein, ein kundenspezifisches Synthesizermodul zu bauen und zu entwerfen, um Chipraum zu sparen. Dagegen sind die Fähigkeiten der „von der Stange erhältlichen" MIDI-Chips für unsere Zwecke allgemein ausreichend, und in der Tat wurde die Nachrichtenübermittlungsmethodologie des Nachrichtenprozessors in Übereinstimmung mit MIDI-Standards entworfen. Eine MIDI-Verarbeitungseinheit könnte somit unsere Spezifikationen mit geringen oder gänzlich ohne Modifikationen erfüllen.

Hardwarekomponenten

[0143] Das HumHorn besteht aus den folgenden Hardwarekomponenten, von denen jede entweder nach Kundenvorstellungen gebaut ist oder „von der Stange" erhältlich ist:

1) ein Gehäuse zum Enthalten aller folgenden Komponenten sowie Batterien;
2) ein Mikrophon;
3) einen oder mehr Lautsprecher;
4) eine Elektronik, die folgende Merkmale umfasst:
a) einen ADC,
b) einen oder mehr Chips zum Ausführen des:
i) SAM,
ii) MP und
iii) SG;
c) einen DRC,
d) einen Verstärker und
e) eine Lautstärkeregelung;
5) mit den Fingern betätigte Steuerungsschalter, Tasten und Skalen; und
6) optional eine kleine Anzeige, um dem Spieler zu helfen, Parameter auszuwählen, und/oder um anzugeben, welche Parameter eingestellt sind.

[0144] All diese Komponenten sind unkompliziert. Lediglich 4)b) erfordert eingehende Überlegungen bezüglich der besten Art und Weise der Implementierung. Drei mögliche Implementierungen sind:

- eine monolithische Lösung, bei der alle drei Verarbeitungselemente auf einem einzigen Chip – ob nach Kundenvorstellungen entworfen oder „von der Stange" – kombiniert sind, der gemäß den oben beschriebenen Spezifikationen programmiert ist;
- zwei separate Chips: einer für das SAM und den MP, der gemäß den oben beschriebenen Spezifikationen programmiert ist; der andere für den SG, wahrscheinlich ein „von der Stange erhältlicher" MIDI-Chip, möglicherweise jedoch ein anderer Mehrzweck-Chip, der für eine Klangsynthese programmiert ist; oder
- ein „von der Stange erhältlicher" MIDI-Chip oder ein anderer Klangsynthesizer, der eine gewisse eingebaute kundenspezifische Programmierung ermöglicht, in den der Code für das SAM und den MP platziert werden.

[0145] In jedem dieser Fälle könnten der ADC, der DAC oder beide bereits auf dem Chip vorliegen. Der Filtermechanismus des SAM könnte durch ein Filtermikrophon oder einen anderen Mechanismus ersetzt werden, der das notwendige Bandpassfiltern mechanisch oder mit einer analogen Schaltungsanordnung durchführt.

[0146] Was die mit den Fingern betätigten Steuerungen betrifft, ist es wünschenswert, zumindest zwei verschiedene Arten zu haben: diejenigen, die sich in eine Position schalten und dort bleiben, und diejenigen, die auf ein Loslassen hin zurückkehren. Als Beispiel betrachte man die für die Tonhöhenachverfolgung verwendeten FACs. Es ist am besten, einen Schalter zu haben, der in einen kontinuierlichen oder diskreten Tonhöhenachverfolgungsmodus versetzt werden kann und der dort bleibt, nachdem er eingestellt wurde. Ferner ist es wünschenswert, eine Taste aufzuweisen, die vorübergehend zu dem entgegengesetzten Modus wechselt. Wenn sich also der Spieler in einem kontinuierlichen Modus befindet und rasch, eine Tonhöhe festlegen möchte oder eine schnelle Tonleiter in einer Musiktonart singen möchte, kann er die Taste drücken und sie dann,

wenn er fertig ist, unmittelbar loslassen. Desgleichen kann der Spieler, wenn er sich in dem diskreten Modus befindet, rasch zu einer anderen Tonhöhe – einschließlich einer außerhalb der aktuellen Musiktonart befindlichen Tonhöhe – gleiten, indem er die Taste vorübergehend drückt und dann durch Loslassen der Taste unmittelbar in die Tonart zurückkehrt. Tasten sind außerdem für ein rasches Wechseln zwischen Instrumenten und Oktaven wünschenswert, was dem Spieler ermöglicht, als Ein-Mann-Band zu fungieren.

[0147] Das Gehäuse des Instruments kann selbst eine Vielzahl von Zwecken und Funktionen aufweisen. Insbesondere kann das Gehäuse zwei Abschnitte aufweisen: einen Innenbehälter und einen Außenmantel. Der Innenbehälter enthält die Elektronik und Batterien in einer einfachen, zweckmäßigen, leicht zu handhabenden und in sich abgeschlossenen Einheit. Sein Zweck besteht darin, die schweren und teuren Posten in einer kompakten und modularen Form zu halten. Der Beitrag des Außenmantels ist seine Gestaltung. Der Außenmantel kann so hergestellt sein, dass er jeder beliebigen traditionellen oder neuartigen Instrumentenform sowohl bezüglich ihrer visuellen als auch ihrer akustischen Eigenschaften ähnelt. Der Mantel kann auch das Mikrophon und/oder (einen) Lautsprecher enthalten. Das Innen- und das Außengehäuse können so hergestellt sein, dass sie ohne weiteres getrennt werden können. Wenn sie ordnungsgemäß aneinander befestigt sind, liefert der Mantel mittels eines physischen Schlüssels auf der Innenoberfläche des Außenmantels, der in einen entsprechenden Schlitz auf der Außenseite des Innenbehälters passt, Informationen an den Innenbehälter. Zusammen mit anderen möglichen Informationen würde der Schlüssel eine Beschreibung des erwarteten Instrumentenklangs liefern, den der SG erzeugen soll. Indem also der Innenbehälter aus einem Mantel in Form eines Instruments gezogen und in einen anderen Mantel in Form eines anderen Instruments eingefügt wird, ändert sich der durch den SG erzeugte Klang von dem des erstgenannten Instruments zu dem des letztgenannten Instruments. Eine Vielzahl unterschiedlicher Mäntel könnte hergestellt werden, damit der Spieler nicht nur den auditiven Eindruck, sondern auch den taktilen und visuellen Eindruck des Spielens eines spezifischen Musikinstruments gewinnen kann.

Halsmikrophon

[0148] Statt dass der Benutzer in ein Mundstück summt, das ein Mikrophon enthält, wie oben erörtert, ist es auch möglich, ein Mikrophon an dem Kragen eines Kleidungsstücks zu platzieren. Dieser Kragen umgibt den Hals, so dass das Mikrophon leicht gegen die Kehle gedrückt wird. Da lediglich die Tonhöhe erfasst wird, ist die Audioqualität anderweitig nicht wichtig – deshalb funktioniert diese Vorgehensweise. Es kann zweckmäßig sein, sich keine Gedanken darum machen zu müssen, ein Mikrophon zu halten oder den Mund in der Nähe eines Mikrophons zu haben.

Trichtermikrophon

[0149] Mundstücke von Instrumenten sind unhygienisch und führen dazu, dass man nur ungern sein Instrument mit jemand anderem gemeinsam benutzt. Durch ein trichterförmiges Behältnis an dem Mikrophonende des Instruments wird man psychologisch und mechanisch davon abgehalten, dasselbe mit seinen Lippen zu berühren. Ferner:

- ermöglicht es eine größere Lippenbewegungsfreiheit, was zum Bilden von Konsonantenlauten wichtig ist, was wichtig beim Erzeugen einer schnellen Sequenz von Einsätzen ist;
- bildet es einen besseren Eingang für den Klang des Singens/Summens seitens des Benutzers; und
- hilft es, den Klang der Stimme des Spielers zu verstecken, so dass Zuhörer sich stattdessen auf den Klang des Instruments konzentrieren können.

Parameterwerte

[0150] Die in der nachfolgenden Tabelle B dargelegten Parameterwerte funktionieren in den durchgeführten Tests gut.

Tabelle B. Parameterwerte

Parameter	Referenz-Position	Bevorzugter Wert
MAX_LENGTH_RATIO (MAX_LÄNGEN_VERHÄLTNIS)	Fig. 8	0,12
w_{max}	Gl. 4	0,02 Sekunden
Δt_d	Gl. 8	0,10 Sekunden
niedrige Grenzfrequenz		100 Hz
hohe Grenzfrequenz		300 Hz
MAX_AMPLITUDE_DIFFERENCE (MAX_AMPLITUDEN_DIFFERENZ)	Fig. 8	40 (beruht auf 8Bit-Abtastwerten)
MAX_DIFFERENCE_WHEN_NOTE_ON (MAX_DIFFERENZ_WENN_TON_EIN)	Fig. 8	0,2
MAX_DIFFERENCE_WHEN_NOTE_OFF (MAX_DIFFERENZ_WENN_TON_AUS)	Fig. 8	0,1
MIN_WAVELENGTH (MIN_WELLENLÄNGE)	Fig. 8	53 Abtastwerte (bei einer Abtastrate von 22.050)

MAX_WAVELENGTH (MAX_WELLENLÄNGE) (dies entspricht w_{max})	Fig. 8	463 Abtastwerte (bei einer Abtastrate von 22.050)
N_SAMPLES_PER_CHECKPOINT (N_ABTASTWERTE_PRO_PRÜFPUNKT)	Fig. 9	11
MAX_VOICE_SPEED (MAX_STIMMEN_GESCHWINDIGKEIT)	Fig. 10	0,2
MIN_ATTACK_GAP (MIN_EINSATZ_ZWISCHENRAUM)	Fig. 14	0,1 Sekunde
MIN_ATTACK_LOUDNESS (MIN_EINSATZ_LAUTHEIT)	Fig. 14	10 (beruht auf 8Bit-Abtastwerten)
MIN_LOUDNESS (MIN_LAUTHEIT)	Fig. 14	8 (beruht auf 8Bit-Abtastwerten)
MIN_TIME_OFF (MIN_ZEIT_AUS)	Fig. 14	0,05 Sekunden
MAX_CONSECUTIVE_WEAK (MAX_AUFEINANDERFOLGEND_SCHWACH)	Fig. 15	5
MAX_PITCH_ERROR (MAX_TONHÖHE_FEHLER)	Fig. 17	0,75 Halbtöne
MAX_BEND_RANGE (MAX_BIEGUNG_BANDBREITE)	Fig. 17	2 Halbtöne
SG_REFRACTORY_PERIOD (SG_REFRAKTÄR_PERIODE)	Fig. 17	0,02 Sekunden

[0151] Die obigen Parameterwerte sind lediglich beispielhaft und sollen keine Einschränkung darstellen. In der eigentlichen Praxis wären andere Parameterwerte gleichermaßen geeignet.

Weitere Fragen

Latenzf'ragen und der „Vorab-Einsatz"

[0152] Für einen Toneinsatz mit einer bisher undefinierten Frequenz weist das oben beschriebene FDM eine Verzögerung von weniger als 30 Millisekunden (etwa 1/30 einer Sekunde) von dem Zeitpunkt, zu dem der Sänger die neue Tonhöhe beginnt, bis sie schließlich durch das FDM erfasst wird, auf. Dabei wird angenommen, dass der tiefste gesungene Ton das C ist, das zwei Oktaven unter dem mittleren C liegt, das eine Frequenz von 65 Hz aufweist (ein außergewöhnlich tiefer Ton), wobei in diesem Fall ein Zyklus 15 Millisekunden dauert und zwei Zyklen 30 Millisekunden dauern. Wenn das SSM einen neuen Instrumenteneinsatz erst erzeugt, nachdem das FDM die Tonhöhe erfasst, kann dieser Einsatz geringfügig wahrnehmbar und möglicherweise misstönend sein, wobei diese Verzögerung betont wird. Es ist möglich, den Eindruck der Verzögerung auf folgende Weise zu verringern. Für jedes Instrument gibt es einen nicht-stimmhaften Einsatzklang. Wenn die Stimme des Sängers eine Schwelle erreicht, beginnt das SSM, den nicht-stimmhaften Einsatzklang zu spielen. Dann, wenn zu dem Zeitpunkt begonnen wird, zu dem das FDM die Tonhöhe erfasst, wird dieser nicht-stimmhafte Klang allmählich in den Klang des Instrumenteneinsatzes bei der erfassten Tonhöhe übergeleitet. Dies würde eine spezialisierte MIDI-Programmierung erfordern, wenn eine standardmäßige MIDI verwendet werden sollte.

Glätten der Tonhöhe

[0153] Oben wurde erwähnt, dass das Instrument aufgrund der geringfügigen Tonhöheschwankungen der Stimme des Sängers, die der Klang des HumHorns in der Tat betonen kann, während einer kontinuierlichen Tonhöhenachverfolgung manchmal weinerlich klingen kann. Es ist möglich, diese Weinerlichkeit beträchtlich abzumildern, indem das resultierende Tonhöheprofil, das durch das Instrument gespielt wird, geglättet wird. Das heißt, dass die Absicht der Tonhöheglättungsfunktion darin besteht, die Flexibilität der kontinuierlichen Tonhöhenachverfolgung zu ermöglichen und dabei die Weinerlichkeit, die manche Instrumente bei den Stimmen mancher Menschen aufweisen, abzumildern.

[0154] Eine Möglichkeit, die Tonhöhe zu glätten, besteht darin, das durch das FDM erzeugte Tonhöheprofil durch ein Tiefpassfilter zu leiten. Ein besseres Verfahren wird erhalten, indem Prinzipien aus der Theorie der Steuersysteme verwendet werden. Man stelle sich die durch das Instrument gespielte Tonhöhe als ein Nachverfolgen des durch das FDM erzeugten Tonhöheprofils vor. Wir können auf die Art, in der dieses Nachverfolgen stattfindet, der Tonhöhe des Instruments Masse verleihen. Insbesondere gilt:

$$E = P_{FDM} - P_{inst}$$

$$d^2 P_{inst}/dt^2 = k1 \cdot E + k2 \cdot \text{int time}(E) - k3 \cdot dP_{in}/dt \qquad (11)$$

wobei P_{FDM} die durch das FDM angegebene Tonhöhe ist, P_{inst} die durch das Instrument zu spielende Tonhöhe ist, E der Tonhöhenachverfolgungsfehler zwischen dem Instrument und der Ausgabe des FDM ist, int_time() für Integration über die Zeit steht und k1, k2 und k3 Konstanten sind. Das Obige ist bei Steuersystemen als Proportional-Integral-Differential-Steuerungsgesetz (PID-Steuerungsgesetz) bekannt und ist eine relativ effektive Art und Weise, wie man das P_{inst} das P_{FDM} glatt nachverfolgen lässt. Der Differentialbegriff (der dritte Begriff) stabilisiert P_{inst} da er einen dämpfenden Effekt aufweist. Er wird verwendet, um Oszillationen in der Steuerung abzudämpfen. Der Integralbegriff verbessert die Genauigkeit der Nachverfolgung. Indem die Werte der Konstanten geändert werden, können wir verschiedene Pegel des Glättens, der Nachverfolgungsgenauigkeit und der Ansprechzeiten erhalten. In der Tat gibt es für diesen Zweck wahrscheinlich bessere Steuerungsgesetze als dieses, z. B. Voreil-Verzögerungs-Steuerung, der Hauptgedanke wird jedoch durch das PID-Steuerungsgesetz dargestellt.

[0155] Wenn ein Tonhöheglättungsverfahren verwendet wird, sollte eine Einrichtung zum automatischen Deaktivieren desselben immer dann vorliegen, wenn eine beabsichtigte Tonänderung oder jegliche Art relativ großer Frequenzänderung vorliegt. Man erinnere sich, dass das Glätten lediglich dazu gedacht ist, die Weinerlichkeit zu beseitigen, die lediglich geringe Frequenzschwankungen (allgemein weniger als einen Halbton) beinhaltet. Diese Strategie kann einfach dadurch implementiert werden, dass $P_{inst} = P_{FDM}$ immer dann eingestellt wird, wenn die Änderung bei P_{FDM} eine bestimmte Schwelle übersteigt.

Netzwerkerweiterungen

[0156] Die folgenden Überlegungen sind auf eine HumBand™-Technologie bezogen, insbesondere die Verwendung von HumBand™ in Bezug auf das Internet, z. B. als Internet-Einrichtung. Der HumBand™-Stimmanalyseprozess extrahiert eine geringe Menge wichtiger Informationen aus dem Stimmstrom und verwendet sie dazu, die gewünschten Instrumente zu spielen. Man schätzt, dass eine nicht-komprimierte Bandbreite von nicht mehr als 300 Bytes/Sekunde notwendig ist, um alle Nuancen zu erfassen, dies kann jedoch ohne Verlust auf geschätzte 500 Bits/Sekunde im Durchschnitt, vielleicht weniger, stark komprimiert werden. Ein drei Minuten dauerndes Lied würde somit etwa 11 Kbytes für eine Stimme verbrauchen. Mehrere Stimmen erfordern proportional mehr. Dies ist eine ziemlich niedrige Zahl und legt nahe, dass HumBand™-Email, -Downloads und andere Formen einer HumBand™-Kommunikation mit geringem Mehraufwand durchgeführt werden können.

Interaktives Musizieren in der Gruppe über Web-/Chat-ähnlichen Dienst

[0157] Um diesen Dienst zu nutzen, wird man ein Mitglied einer Online-Gruppe, nachdem man sich mit Name und Passwort in die Website HumJam.com eingeloggt hat. Jede Person in der Gruppe ist zu einer bestimmten Zeit entweder Teil des Publikums bzw. Zuhörer oder ein Musizierender.

Zuhörer:

[0158] Als Zuhörer kann man eine Darbietung während der Darbietung in Echtzeit kommentieren und erör-

tern. Es kann spezielle Symbole oder Hör-Icons geben, die eine bestimmte Bedeutung haben und die dem Darbietenden gesendet werden können. Beispielsweise Applaus, Bravo-Rufe, Buh-Rufe, Gelächter, Beifallsrufe und Pfeifen, die bzw. den bzw. das der Darstellende hört. Ferner kann jeder Angehörige des Publikums an einer Wähl- bzw. Abstimmrunde teilnehmen, um seine subjektive Meinung bezüglich der Qualität der Darbietung auszudrücken.

Darbietender:

[0159] Der Darbietende wird aufgrund seines angeborenen versteckten Wunsches, live vor einem Publikum zu spielen, zu der Session hingezogen. Das ist aufregend und macht Spaß, und aufgrund der Anonymität des Internets sowie der durch die HumBand™ gelieferten stimmlichen Tarnung ist sie auch weniger abschreckend als eine Darbietung auf der Bühne. Man stelle sich vor, eine Darbietung für eine Menge von Dutzenden oder Hunderten (oder sogar Tausenden!) in der abgeschiedenen Privatheit seines eigenen Heims zu geben. Während der Darbietung ist das HumBand™-Instrument über eine Schnittstelle direkt mit dem Internet verbunden, so dass die Darbietung über die Website HumJam.com live übertragen werden kann. Der Darbietende bekommt live eine Rückmeldung von den Zuhörern, und am Ende der Darbietung kann er eine Bewertung durch die Zuhörer empfangen.

[0160] Wählen bzw. Abstimmen wird zu drei Zwecken durchgeführt

- Um das Niveau der Darbietungsgruppe zu erhöhen/verringern. Wenn die Rangfolge einer Person (durch Wählen bzw. Abstimmen) auf ein ausreichend hohes Niveau angestiegen ist, kann man in einer Darbietungsgruppe auf einem höher bewerteten Niveau mitmachen. Auf diesem Niveau spielt man für ein Publikum von Darbietenden, die sich in der gleichen Rangfolge befinden. Beispielsweise soll das Eintrittsniveau Rang 1 sein. Bei Rang 1 kann jeder spielen, und jeder kann wählen bzw. abstimmen. Diejenigen mit ausreichend hohen Stimmzahlen können zu Rang 2 aufsteigen. Bei Rang 2 wird man nur von anderen beurteilt, die Rang 2 oder einen höheren Rang erreicht haben.
- Um den Rang und das Gewicht beim Wählen bzw. Abstimmen zu erhöhen/verringern. Wenn eine Person, die einen höheren Rang aufweist als der Darbietende, über die Darbietung abstimmt, zählt seine Stimme mehr als die Stimmen derjenigen, deren Rang niedriger ist.
- Um die Chancen, einen Preis zu empfangen, zu erhöhen/verringern. Dem Darbietenden, der für diesen Zeitraum den höchsten Rang aufweist, oder möglicherweise dem Darbietenden, dessen Rangfolge sich über einen gegebenen Zeitraum hinweg am stärksten erhöht hat, können regelmäßig, z. B. täglich/wöchentlich/monatlich, Preise verliehen werden. Dann ähnelt eine Rangfolge insofern einer Vorgabe beim Golfspielen, als die Chance, einen Preis zu gewinnen, davon abhängt, wie gut die eigene Leistung in Bezug auf das bisherige oder durchschnittliche Können ist.

[0161] Das obige Verfahren ist ein Vorschlag einer einzelnen Art eines interaktiven Szenarios, das an die Lust vieler Menschen am Wettbewerb und an Darbietungen appellieren könnte. Ferner wäre eine Art Preis die Gelegenheit, für ein sehr großes Publikum, z. B. die zusammengefassten Zuhörerschaften aller Gruppen, oder auf speziell beworbenen Veranstaltungen zu spielen, auf denen die gewinnenden Darbietenden (jeglicher Rangfolge) spielen.

Ein internationales Unterfangen.

[0162] Derartige interaktive Darbietungen wären eines der einzigen Online-Beispiele für echte, ungehinderte internationale Kommunikationen, da Musik kulturelle/sprachliche Schranken durchbricht. Das Internet und HumBand™ könnten eine bisher ungekannte Form einer unmittelbarer internationaler Kommunikation einleiten.

Technische Fragen

[0163] Jeder Darbietende und jeder Zuhörer ist in der Lage, über seine Internet-fähige HumBand™ teilzunehmen. Alle Darbietenden senden Informationen mittels ihrer HumBand™. Alle Zuhörer hören bei diesen Darbietungen über ihre HumBands™/PCs/PC-Kopfhörer/HumBand™-Kopfhörer oder eine andere HumBand™-Codec-befähigte Vorrichtung zu.

[0164] Der Darbietende spielt zu einer Begleitung, die durch den HumJam.com-HumServer™ bereitgestellt wird. Der Server sendet die Begleitungsinformationen über den HumBand-Codec an den Darbietenden. Die Begleitung wird auf jeglicher freigegebenen Vorrichtung gespielt. Der HumBand-Codec ähnelt sehr stark MIDI, ist jedoch vielleicht bezüglich einer Stimmsteuerung optimiert.

[0165] Der Darbietende spielt synchron mit seiner Begleitung, und sein Signal wird über diesen selben Codec an den Server zurückgesendet. Erst dann sendet der Server die Darbietung mittels Rundsendung an das Publikum. Für das Publikum sind der Darbietende und die Begleitung in perfekter Synchronie. Es gibt kein Latenzproblem. Dies ist darauf zurückzuführen, dass der Server das Signal des Darbietenden empfängt und in der Lage ist, es mit der Begleitung zu kombinieren, und zwar zeitlich ordnungsgemäß gesteuert, so dass das resultierende Signal die Darbietung, wie sie durch den Darbietenden gehört wird, reproduziert. Obwohl also eine geringe Verzögerung vorliegt, wird die Darbietung somit dennoch live und in voller Wiedergabetreue rundgesendet.

[0166] Zuhörer senden ihre Kommentare und Stimmabgaben an den Server, der sie nachzählt, organisiert und aufteilt.

Jamsessions für mehrere Darbieter

[0167] Dieses Szenario wird am meisten durch das Latenzproblem behindert. Insbesondere liegt eine merkliche Zeitverzögerung zwischen einem Senden eines Signals seitens eines Darbietenden über das Internet und der Ankunft desselben vor. Für die meisten Formen der Kommunikation ist diese begrenzte Latenz nicht nachteilig, da eine Synchronie nie erforderlich ist. Eine Verzögerung von 200 ms zwischen dem Senden einer Kommunikation und deren Empfang ist kaum wahrnehmbar. Wenn jedoch mehrere Teilnehmer versuchen, einen Klang zu synchronisieren, macht eine derartige Verzögerung dies unmöglich. Jeder Darbietende wartet auf das Signal des anderen, um zu synchronisieren. Diese Verzögerung verstärkt zusätzlich die Verzögerung an dem anderen Ende. Der Effekt vervielfacht sich, und es kann keine Form der Synchronie aufrechterhalten werden.

[0168] Um diesen Kaskadeneffekt (Schneeballeffekt) zu eliminieren und das allgemeine Latenzproblem abzumildern, es jedoch nicht gänzlich zu eliminieren, kann ein zentraler Server – ein Leiter – einen stetigen Puls, z. B. das Ticken eines Metronoms, an alle Parteien senden, der bzw. das zeitlich so gesteuert ist, dass jeder Teilnehmer das Signal gleichzeitig empfängt. Jeder Darbietende kann seine Darbietung dann zeitlich auf diesen Puls abstimmen und eine geringfügige Verzögerung von den anderen Darbietenden erwarten, wobei er allmählich (oder vielleicht schnell) lernt, diese geringe Verzögerung zu ignorieren und sich an diese anzupassen. Der Puls ist effektiv die Begleitung. Eine Software an dem Ort des Darbietenden kann diese Verzögerung nach Abschluss des Stücks berücksichtigen und kann für jeden Darbietenden den Klang der gesamten Darbietung ohne die Verzögerungen abspielen.

Gruppenkompositionen

[0169] Die obigen Konzepte können kombiniert werden, um ein Abstimmen über einzelne musikalische Parts zu ermöglichen, die anschließend miteinander verbunden werden, um eine aus einer Gruppenanstrengung hervorgegangene HumBand™-Komposition zu erzeugen.

Song-E-Mail, „HumMail™"

[0170] Man kann aus einer oder aus mehreren Stimmen bestehende Songs, die man auf seiner HumBand™ aufgezeichnet hat, aufzeichnen und per E-Mail an Freunde versenden, die die Songs auf ihren eigenen HumBands™ abspielen können.

Herunterladen von Songs

[0171] Es können Darbietungen von der Site heruntergeladen werden, unterschiedliche Stücke, die von verschiedenen interessanten und/oder berühmten Darbietenden gespielt werden. Da die heruntergeladenen Informationen viel präziser sind, da sie eine weitaus größere Feinheit und Nuance enthalten als übliche MIDI-Darbietungen, kann man einen größeren Realismus und eine größere ansprechende Wirkung erwarten.

Downloads von Begleitungen

[0172] Die Begleitungsabschnitte vieler vieler unterschiedlicher Stücke könnten zur Verfügung gestellt werden, so dass man sie in seine HumBand™ herunterladen könnte (niedrige Bandbreite) und dann dazu spielen könnte.

Kompositions-Kettenbriefe

[0173] Genauso wie man eine Begleitung herunterladen kann, kann man eine Begleitung per E-Mail empfangen, ein Musikstück anhängen und das Ganze weitersenden, eine Art „serielles Jamming". Alternativ dazu könnten unvollständige Stücke zu der Site hochgeladen werden, um zu ermöglichen, dass andere mitspielen und möglicherweise auch etwas beitragen.

Selbstbegleitung

[0174] Auf dem Markt gibt es ein Softwareprodukt, das eine automatische Begleitung für Instrumente und Stimme liefert. Der Darbietende wählt ein Stück, und die Software spielt den Hintergrund und folgt gleichzeitig der Stimme/dem Instrument des Darbietenden. Die Geschwindigkeit der Begleitung wird kontinuierlich modifiziert, um auf die Geschwindigkeit des Darbietenden abgestimmt zu sein. Ein ähnliches System könnte für die HumBand™ gebaut werden, vielleicht im Bunde mit dieser Firma, das dem Darbietenden folgt und eine intelligente Begleitung liefert. Diese Begleitungen könnten von der Website heruntergeladen werden.

Kontextsensible Tonhöhekorrektur

[0175] Es könnten Song-Partituren in die HumBand™ heruntergeladen und, der Stimme des Darbietenden folgend, abgespielt werden, wobei die tatsächliche Stimme mit der beabsichtigten Musik abgestimmt wird. Sogar Spieler mit einer sehr schlechten Tonhöhekontrolle könnten absolut tonrein spielen. Dies könnte beinhalten, dass Instrumenten- und Oktavwechsel an bestimmten Stellen in dem Song automatisch erfolgen.

Downloads von Instrumenten-Wavetables

[0176] Wenn neue Instrumentenklänge (sowohl natürliche als auch synthetische) verfügbar werden, können ihre Wavetables an die Website gesendet werden. Somit können HumBand™-Besitzer zu jeder Zeit auf eine riesige Bibliothek interessanter Instrumentenklänge aus dem Internet zugreifen, und ihre Auswahl möglicher Klänge ist im Vergleich zu den wenigen, die zur Zeit des Kaufs auf der HumBand™ installiert sind, stark erweitert. Die für einen Wavetable-Download erforderlichen Daten sind nicht so vernachlässigbar wie für einen Download eines Songs.

Downloads von Steuerungssoftware

[0177] Software, „Humlets™, könnten von der Site in die HumBand™ herunterladbar sein und könnten auf viele sinnvolle Arten eine Steuerung modifizieren, z. B.:

- Hinzufügen/Verbessern von Effekten, z. B. Echoplex, Chorus und andere Verzerrungen.
- Hinzufügen von Möglichkeiten, z. B. zusätzliche Ausschmückungen/Riffe, die der Spieler willentlich aufrufen kann. Diese Riffe/Ausschmückungen können kontextsensibel sein, so dass sie je nach Skalengrad, Rhythmus und Stil des Stücks zum Zeitpunkt des Aufrufs auf unterschiedliche Weise gespielt werden können. Der Stil des Stücks könnte zusammen mit dem Download ausgewählt werden. Beispiele sind: Barock, Klassik, Jazz, Dixieland und Hardrock. Diese Kontextsensibilität könnte gemäß einer im Voraus (in Hardware) eingerichteten bestimmten Formel angewendet werden oder könnte in einer heruntergeladenen Software verkörpert sein.
- Hinzufügen einer neuen Funktionalität, die beispielsweise die Art und Weise, auf die kontextsensible Effekte und Begleitungen gespielt werden, modifizieren/verbessern könnte.
- Modifizieren der Kernfunktionalität. Wenn sich unsere Tonhöheerkennungs- und Einsatzerfassungsalgorithmen verbessern, könnten diese Verbesserungen als Updates herunterladbar sein.
- Neue Kernfunktionalität. Die absoluten Grundlagen der HumBand™ könnten modifizierbar sein. Beispielsweise könnte eine neue Funktionalität hinzugefügt werden, z. B. eine Klangfarbensteuerung durch die Erkennung bestimmter stimmlicher Nuancen, z. B. Veränderungen von Oberschwingungen der Stimme.

Unterricht

[0178] Ebenfalls von der Website HumJam.com erhältlich: Musikunterricht. Unterricht könnte von Menschen bereitgestellt werden (bezahlt) oder von Software (kostenlos). Software könnte online oder offline verwendet werden und würde dem Lernenden beispielsweise dabei helfen, die Tonhöhenkontrolle zu verbessern. Dies könnte erfolgen, indem eine Auswahl für den Lernenden gespielt wird oder indem dem Lernenden alternativ dazu ermöglicht wird, eine Partitur zu lesen, und indem darauf gewartet wird, dass der Lernende spielt, was er gehört oder gelesen hat. Die Software könnte sowohl die richtige Tonhöhe als auch die Tonhöhe des Lernen-

den als zwei gleichzeitige Echtzeitgraphen zeigen. Der Lernende könnte dann visuell sehen, wenn seine Tonhöhe zu hoch oder zu tief ist.

Besonders hohe Wiedergabetreue

[0179] Wenn die Darbietungsfähigkeiten der HumBand™ des Spielers unzureichend sind (je nach dem Modell), kann er HumMail™ an die Website senden und im Gegenzug eine komprimierte Version der Aufzeichnung empfangen, z. B. im MP3-Format, die auf der besten erhältlichen HumBand™-Ausrüstung durchgeführt wurde. Da dies ein etwas umständlicher Vorgang wäre, würde kein Spieler dies als selbstverständliche Sache tun, es wäre jedoch in den Fällen ein sinnvoller Dienst, in denen der Darbietende eine besonders gute Aufnahme polieren und behalten möchte. Am wichtigsten ist, dass dies eine gute Werbefläche für die Ausrüstung am oberen Ende der Bandbreite liefern würde und dadurch die Verkaufszahlen anheben würde.

Lernspiele

[0180] Die Site HumJam.com könnte HumBandTM-Spiele zur Online-Nutzung oder zum Herunterladen sponsern. Ein Beispiel ist ein „Simon"-artiges Spiel, bei dem der Spieler eine Abfolge von Tönen nachahmen muss. Wenn der Spieler die Töne korrekt wiederholt, wird die Sequenz um einen anderen Ton erweitert. Zusätzlich dazu, dass er die Töne einfach in der richtigen Reihenfolge singt, muss der Spieler sie vielleicht auch mit unterschiedlichen Instrumentenklängen spielen, die zwischen Tönen wechseln, oder er muss sie vielleicht in wechselnden Oktaven spielen.

Infrarot-Vernetzung

[0181] Die HumBand™ könnte mit einem Infrarottor gebaut werden, um eine drahtlose Vernetzung zwischen Instrumenten, die sich im selben Raum befinden, zu ermöglichen. Mögliche Verwendungen sind:

- Gemeinsam verwendete Lautsprecher. Die HumBand™ jedes Spielers könnte die Ausgabe von allen teilnehmenden Spielern spielen, wobei jeder Person ermöglicht wird, durch seine eigene HumBand™ die ganze Gruppe zu hören.
- Stille Jamsessions. Jeder Spieler schließt Kopfhörer an seine eigene HumBand™ an und kann alle Instrumente spielen hören. Beobachter hören nur ein Bündel von Summtönen. Geeignet für lange Nächte und empfindliche Nachbarn.
- Synchronisation. Die HumBand™ könnte sich automatisch an einen Beat anpassen, der durch eine zentrale Quelle vorgegeben ist oder über die diversen Spieler gemittelt ist, um Einsätze und Aufhebungen erkennen und erzeugen zu helfen. Dies würde unter Anfängern einen besser koordinierten Klang fördern, z. B. bei Schulkindern in HumBand™-Chorgruppen.
- Korrektur der Gruppentonhöhe. Dasselbe Konzept wie oben kann auf die Regulierung der Tonhöhe angewendet werden. Die Tonhöhekorrekturalgorithmen können dazu ausgelegt sein, den durchschnittlichen Spieler bestmöglich zu berücksichtigen, wobei sie die Tonhöhe von Ausreißern automatisch so anpassen, dass sie mit dem Rest der Band übereinstimmen.
- Eigenständiges Spiel. Mehrstimmige Stücke, die vorab aufgezeichnet wurden, können auf einer Gruppe von Instrumenten abgespielt werden, wobei jedes eine einzelne Stimme nimmt und sie zeitlich auf die anderen Instrumente abgestimmt spielt. Eine Gruppe von Instrumenten, die bedienungsfrei dasteht, könnte einen unheimlichen Eindruck einer unsichtbaren Band vermitteln. Oder bei richtiger Durchführung kann vor den Augen des Publikums eine eindrucksvolle Aufzeichnung ins Leben gerufen werden. Dies ist keine interaktive Anwendung der HumBand™, aber Freunden und Nachbarn ist dies vielleicht nicht klar ...
- Jam-Darbietungen vor Ort mit HumJam.com. Örtliche Jamsessions unter Gruppen von Freunden, die zusammen im selben Raum sitzen, könnten durch die HumJam.com-Website rundgesendet werden. Infrarot-Vernetzung, vernachlässigbare Latenz und eine niedrige Bandbreite ermöglichen, dass das LAN-Band die gesamte Darbietung live im Internet ausstrahlt.
- Automatische Teile. Da viele Töne gleichzeitig gespielt werden, kann die Harmonie abgeleitet werden, und neue musikalische Teile können automatisch erfunden und zusammen mit dem Rest der Band gespielt werden: Der AI-Improvisierende.
- Technisch ausgefeilte Interaktionen. Es besteht das Potential eines neuen Paradigmas einer musikalischen Zusammenarbeit, die entstehen kann, wenn alle Instrumente in der Band sowohl in Echtzeit verbunden werden als auch mit einem sehr hohen Steuerungsniveau bedient werden. Dies ist eine grundlegend andere Art von Zusammenspiel als es heutzutage in Bands existiert . Die Möglichkeiten einer Interaktion scheinen endlos, z. B.:

  Durch einen Dirigenten gesteuerte Darbietungen. Man stelle sich eine Band vor, bei der der Dirigent die Darbietung nicht nur indirekt durch händische Gesten, sondern direkt steuert, indem er die Instrumenten-

klänge jeder Gruppe dynamisch auswählt! Andere Parameter, z. B. Tonhöhe, Oktave und Lautstärke, könnten ebenfalls gesteuert werden.
Die Avantgarde. Durch die Verwendung interaktiver Dynamik sorgen moderne Komponisten oft für zusätzliche Willkür und Komplexität bei Darbietungen. Jede HumBand könnte dazu programmiert sein, derart mit den anderen zu kooperieren, dass ein Spielen bestimmter Töne oder ein Auswählen bestimmter Parameter an einem Instrument die Parameter an einem anderen Instrument automatisch verändert. Ein organischer Synergismus könnte sich entwickeln, wenn die Spieler selbst von der fortwährenden Veränderlichkeit der Darbietung ebenso überrascht sind wie das Publikum.

Verwendung eines adaptiven Moduls bei dem FDM

[0182] Die Zuverlässigkeit des FDM kann durch die Hinzufügung eines adaptiven Moduls, z. B. eines neuralen Netzwerks, beträchtlich verbessert werden. In der Tat eignet sich die Architektur für das FDM sehr gut für diese Verbesserung. Der vorliegende Lösungsansatz besteht darin, das adaptive Modul in die Wellenformvergleichsroutine zu integrieren, um den Code im Kästchen **63** im Wesentlichen zu ersetzen. Fig. 33 zeigt ein neurales Vorwärtsregelungs-Mehrschicht-Perzeptronnetzwerk (neurales FFMLP-Netzwerk, FFMLP = feedforward multilayer perceptron), dessen Eingaben die Abtastwerte bei gleichmäßig beabstandeten Intervallen entlang der Wellen sind. Die erste Wellenform (zwischen t_{start} und $t_{aufteilen}$) wird in die Eingangsschicht **334** eingegeben, und Abtastwerte von der Welle zwischen $t_{aufteilen}$ und $t_{aktuell}$ werden in deine Eingangsschicht **336** eingegeben. Verbindungen von den Eingangsschichten werden anschließend zu der verborgenen Schicht **332** weitergeleitet, und Verbindungen von dieser Schicht werden zu dem einzigen Ausgangsknoten **338** weitergeleitet. Die gewünschte Ausgabe des Letzteren, d. h. des Netzwerks, ist die Wahrscheinlichkeit, dass die Form der ersten Welle mit der Form der zweiten Welle übereinstimmt. Die Variable „Differenz" (siehe Fig. 9) ist so definiert, dass sie auf eine gewisse Weise invers mit dieser Wahrscheinlichkeit in Beziehung steht.

[0183] Es gibt viele Arten, die Eingabe in das Netzwerk vorab zu verarbeiten, und es gibt viele Arten von adaptiven Modulen, die denselben Zweck erfüllen könnten. Ferner kann ein Netzwerk oder ein anderer adaptiver Algorithmus gebaut werden, um sowohl das Kästchen **62** als auch das Kästchen **63** zu ersetzen. Der Algorithmus nimmt eine spezifische Anzahl der jüngsten Abtastwerte als seine Eingabe, und der Algorithmus ist trainiert, an seinem Ausgang eine Schätzung für den Aufteilungspunkt innerhalb dieser Werte zu erzeugen. Jedoch besteht der Vorteil des FDM darin, dass im Vergleich zu standardmäßigen Autokorrelations-Lösungsansätzen ein viel kleinerer Satz von Wellen auf eine Übereinstimmung hin getestet wird. Das FDM ist somit sogar dann immer noch effizient, wenn ein komplexeres und adaptiveres Formvergleichsmodul das Kästchen **63** ersetzt.

[0184] Eine sehr populäre Art und Weise, ein FFMLP zu trainieren, beinhaltet einen Rückausbreitungsalgorithmus. Wie bei jedem überwachten Lernverfahren erfordert das Netzwerk Beispiele erwünschter Ausgaben für gegebene Eingaben. Eine große Sammlung von Klangdateien (SFs – sound files) muss von einem repräsentativen Teilsatz der erwarteten Benutzerpopulation erhalten werden, und diese Dateien werden anschließend markiert, wo es angebracht ist, um zu benennen, welche die gewünschten Eingaben und Ausgaben sein sollen. Beispielsweise könnte jede Klangdatei eine begleitende Markierungsdatei aufweisen, die Tripletts der Form t_{start}, $t_{aufteilen}$ und $t_{aktuell}$ enthält, die Versätze in der Wellendatei benennen, zusammen mit Zielwerten für das Kästchen **108**. Der Zielwert ist gleich Eins, wenn die Segmente übereinstimmen, und Null, wenn sie nicht übereinstimmen. Diese Art überwachten Trainings für FFMLPs kennt und versteht man ziemlich gut. Alternativ dazu könnte der Benutzer eine Rückmeldung direkt an das Netzwerk liefern, wenn die durch das HumHorn erzeugte Tonhöhe falsch ist. In beiden Fällen passt sich das Netzwerk an, um die Fehlerwahrscheinlichkeit in ähnlichen Situationen in der Zukunft zu verringern.

[0185] Obwohl die Erfindung hierin unter Bezugnahme auf das bevorzugte Ausführungsbeispiel beschrieben wird, werden Fachleute ohne weiteres erkennen, dass die hierin dargelegten Anwendungen durch andere Anwendungen ersetzt werden können, ohne von dem Schutzumfang der vorliegenden Erfindung abzuweichen. Demgemäß sollte die Erfindung lediglich durch die nachfolgend enthaltenen Patentansprüche beschränkt sein, so dass lediglich die unabhängigen Patentansprüche die Erfindung definieren.

Patentansprüche

1. Ein in der Hand zu haltendes, in sich abgeschlossenes, sprachgesteuertes elektronisches Musikinstrument, das ein Mundstück (**5**), an dem die Stimme eines Benutzers eintritt, ein Signalanalysemodul (SAM, **33**), das ein Eingangssignal (**40**) von dem Mundstück (**5**) empfängt, eine oder mehrere Benutzersteuerungen (FAC, **1a**, **1b**, **37**), eine oder mehrere Klangwiedergabevorrichtungen (**34**, **35**, **36**), die mit dem Signalanalysemodul

(SAM, **33**) gekoppelt sind, ein Gehäuse (**11**), wobei das Mundstück (**5**), die eine oder die mehreren Benutzersteuerungen (FAC, **1a**, **1b**, **37**) und die eine oder die mehreren Klangwiedergabevorrichtungen (**34**, **35**, **36**) gänzlich innerhalb der Grenzen des Gehäuses (**11**) enthalten sind, aufweist und das ferner folgende Merkmale aufweist

ein Klangsynthetisierungsmodul (SSM, **38**), das mit dem Signalanalysemodul (SAM, **33**) gekoppelt und gänzlich innerhalb der Grenzen des Gehäuses (**11**) enthalten ist; und

wobei das Signalanalysemodul (SAM, **33**) die Frequenz des Eingangssignals (**40**) identifiziert und ein Frequenzausgangssignal (**42**) liefert, das dieselbe angibt, und die Lautheit des Eingangssignals (**40**) bestimmt und ein Lautheitsausgangssignal (**43**) liefert, das dieselbe angibt;

wobei das Klangsynthetisierungsmodul (SSM, **38**) sowohl das Frequenzausgangssignal (**40**) als auch das Lautheitsausgangssignal (**43**) von dem Signalanalysemodul (SAM, **33**) empfängt, wobei das Klangsynthetisierungsmodul (SSM, **38**) das Frequenzausgangssignal (**40**) in ein Tonhöhesignal transformiert (**162**) und das Lautheitsausgangssignal (**43**) in einen Volumensteuerwert umwandelt (**163**), um ein Ausgangsprobensignal zu erzeugen, das im Wesentlichen sowohl die Tonhöhe als auch das Volumen der Stimme des Benutzers erfasst;

wobei die eine oder die mehreren Klangwiedergabevorrichtungen (**34**, **35**, **36**) das Ausgangsprobensignal empfangen und einen Ausgangsklang (**13**) erzeugen, der im Wesentlichen sowohl die Tonhöhe als auch das Volumen der Stimme des Benutzers erfasst und den Klang eines traditionellen Instruments imitieren kann;

wobei die Tonhöhe und das Volumen des Instruments im Wesentlichen der Tonhöhe und dem Volumen der Stimme des Benutzers folgen.

2. Das Instrument gemäß Anspruch 1, bei dem das Signalanalysemodul (SSM, **38**) folgendes Merkmal aufweist:

eine Tonhöheerfassungstechnik, um die Frequenz des Eingangssignals (**40**) für das sprachgesteuerte elektronische Musikinstrument zu identifizieren.

3. Das Instrument gemäß Anspruch 2, bei dem die Tonhöheerfassungstechnik folgende Schritte umfasst:

Bestimmen kürzlich erfolgter Zeitschritte;

Summieren von Unterschieden über eine kleine Teilmenge von kürzlich erfolgten Zeitschritten, um eine richtige Grundfrequenz zu finden;

Bestimmen von Wellensegmenten; und

Vergleichen lediglich von Wellensegmenten, die durch Wellenformmerkmale wie z.B. Spitzen oder Nulldurchgänge begrenzt sind.

4. Das Instrument gemäß Anspruch 1, bei dem das Gehäuse (**11**) in einer Form gebildet ist, die ein Musikinstrument darstellt.

5. Das Instrument gemäß Anspruch 1, bei dem das Signalanalysemodul (SAM, **33**) ferner folgende Merkmale aufweist:

ein Frequenzerfassungsmodul (FDM, **45**) zum Empfangen des Eingangssignals (**40**), zum Identifizieren der Frequenz des Eingangssignals und zum Liefern des Frequenzausgangssignals (**42**), das dieselbe angibt, an das Klangsynthetisierungsmodul (SSM, **38**);

ein Lautheitnachverfolgungsmodul (LTM, **47**) zum Empfangen des Eingangssignals (**40**), zum Bestimmen der Lautheit des Eingangssignals (**40**) und zum Liefern des Lautheitsausgangssignals (**43**), das dieselbe angibt, an das Klangsynthetisierungsmodul (SSM, **38**); und

ein Toneinsatzmodul (PADM, **46**) zum Empfangen des Eingangssignals (**40**), zum Bestimmen eines Ton-Ein-/-Aus-Werts und eines Einsatzwerts und zum Liefern eines Ton-Ein-/-Aus-Ausgangssignals (**41**) und eines Einsatzausgangssignals (**44**), die dieselben angeben, an einen Klangsynthetisierer (SSM, **38**).

6. Das Instrument gemäß Anspruch 1, bei dem zumindest eine der einen oder der mehreren Benutzersteuerungen (FAC, **37**, **1a**, **1b**) entweder eine Steuerung, die auf eine Betätigung eines Benutzers hin umschaltet und in ihrer Position verbleibt, oder eine Steuerung, die auf eine Freigabe durch einen Benutzer hin zurückkehrt, umfasst.

7. Das Instrument gemäß Anspruch 1, bei dem das Klangsynthetisierungsmodul (SSM, **38**) auf steuerbare Weise wirksam ist, um entweder eine kontinuierliche Tonhöhenachverfolgung oder eine diskrete Tonhöhenachverfolgung zu liefern (**162**), wobei das Tonhöhesignal der exakten Frequenz des Frequenzausgangssignals (**40**) während der kontinuierlichen Tonhöhenachverfolgung entspricht, und wobei das Tonhöhesignal dem Ton entspricht, der während der diskreten Tonhöhenachverfolgung am nächsten bei dem Frequenzausgangssignal (**40**) liegt; und wobei zumindest eine der einen oder der mehreren Benutzersteuerungen (FAC, **37**, **1a**, **1b**) einen Schalter umfasst, der auf wählbare Weise entweder die kontinuierliche Tonhöhenachverfolgung oder die

diskrete Tonhöhenachverfolgung einstellen kann, wobei der Schalter bei der ausgewählten Tonhöhenachverfolgung bleibt, nachdem dieselbe eingestellt wurde.

8. Das Instrument gemäß Anspruch 1, bei dem das Klangsynthetisierungsmodul (SSM, **38**) auf steuerbare Weise wirksam ist, um entweder eine kontinuierliche Tonhöhenachverfolgung oder eine diskrete Tonhöhenachverfolgung zu liefern (**162**), wobei das Tonhöhesignal der exakten Frequenz des Frequenzausgangssignals (**40**) während der kontinuierlichen Tonhöhenachverfolgung entspricht, und wobei das Tonhöhesignal dem Ton entspricht, der während der diskreten Tonhöhenachverfolgung am nächsten bei dem Frequenzausgangssignal (**40**) liegt; und wobei zumindest eine der einen oder der mehreren Benutzersteuerungen (FAC, **1a**, **1b**, **37**) eine Taste (FAC, **1a**, **1b**, **37**) umfasst, die das Instrument vorübergehend von der kontinuierlichen Tonhöhenachverfolgung zu der diskreten Tonhöhenachverfolgung wechseln lässt, wobei, wenn der Benutzer das Instrument in der kontinuierlichen Tonhöhenachverfolgung bedient und eine Tonhöhe auswählen oder eine Tonleiter in einem Musikmodus singen möchte, der Benutzer die Taste (FAC, **1a**, **1b**, **37**) drücken und, wenn er fertig ist, unverzüglich loslassen kann.

9. Das Instrument gemäß Anspruch 1, bei dem das Klangsynthetisierungsmodul (SSM, **38**) auf steuerbare Weise wirksam ist, um entweder eine kontinuierliche Tonhöhenachverfolgung oder eine diskrete Tonhöhenachverfolgung zu liefern (**162**), wobei das Tonhöhesignal der exakten Frequenz des Frequenzausgangssignals (**40**) während der kontinuierlichen Tonhöhenachverfolgung entspricht, und wobei das Tonhöhesignal dem Ton entspricht, der während der diskreten Tonhöhenachverfolgung am nächsten bei dem Frequenzausgangssignal (**40**) liegt; und bei dem zumindest eine der einen oder der mehreren Benutzersteuerungen (FAC, **1a**, **1b**, **37**) eine Taste umfasst, mittels derer ein Benutzer, während das Instrument für die diskrete Tonhöhenachverfolgung eingestellt wird, vorübergehend zu einer anderen Tonhöhe ziehen kann, einschließlich einer außerhalb eines aktuellen Musikmodus liegenden Tonhöhe, indem er vorübergehend die Taste drückt.

10. Das Instrument gemäß Anspruch 1, bei dem zumindest eine der einen oder der mehreren Benutzersteuerungen (FAC, **1a**, **1b**, **37**) eine Taste zum Wechseln zwischen Oktaven umfasst.

11. Das Instrument gemäß Anspruch 1, bei dem das Gehäuse (**11**) folgende Merkmale umfasst:
einen Innenbehälter und einen Außenmantel;
wobei der Innenbehälter das Signalanalysemodul (SAM, **33**) enthält; und
wobei der Außenmantel so hergestellt ist, dass er einem traditionellen Musikinstrument ähnelt.

12. Das Instrument gemäß Anspruch 11, bei dem der Außenmantel das Mundstück (**5**) und/oder die eine oder die mehreren Klangwiedergabevorrichtungen (**34**, **35**, **36**) enthält.

13. Das Instrument gemäß Anspruch 1, das ferner folgendes Merkmal aufweist:
eine Stromquelle, die gänzlich innerhalb der Grenzen des Gehäuses (**11**) enthalten ist.

14. Das Instrument gemäß Anspruch 11, bei dem der Innenbehälter und der Außenmantel so gebildet sind, dass sie zusammenpassen, wobei der Außenmantel eine Instrumentbeschreibung und einen Kommunikationsweg umfasst, der Informationen an den Innenbehälter liefert, die eine Beschreibung eines tatsächlichen Instrumentenklanges umfassen, der durch ein traditionelles Instrument erzeugt wird, dem der Außenmantel ähnelt; bei dem ein Herausziehen des Innenbehälters aus dem Außenmantel, wobei der Außenmantel in Form eines Instruments erzeugt ist, und ein Einfügen desselben in einen anderen Außenmantel, der in Form eines anderen Instruments vorgesehen ist, den Innenbehälter dazu konfiguriert, einen Klang zu liefern, der durch ein Instrument erzeugt wird, dem der Außenmantel, mit dem der Innenbehälter derzeit ein Paar bildet, ähnelt.

15. Das Instrument gemäß Anspruch 14, bei dem ein physischer Schlüssel auf einer Innenoberfläche des Außenmantels in einen entsprechenden Schlitz auf einer Außenseite des Innenbehälters passt, um das Instrument dazu zu konfigurieren, einen Klang wiederzugeben, der einem tatsächlichen Instrument zugeordnet ist, das die Form aufweist, die durch den Außenmantel dargestellt wird.

16. Das Instrument gemäß Anspruch 14, das ferner folgendes Merkmal aufweist:
eine Mehrzahl unterschiedlicher Außenmäntel, die so hergestellt sind, dass der Benutzer nicht nur den auditiven Eindruck, sondern auch einen taktilen und visuellen Eindruck erhält, ein spezifisches Musikinstrument zu spielen.

17. Das Instrument gemäß Anspruch 1, bei dem das Gehäuse (**11**) in Form entweder einer Trompete, einer Posaune, eines Saxophons, einer Oboe, eines Fagotts, einer Klarinette, einer Flöte, eines Klaviers, einer elek-

trischen Gitarre oder einer Pfeife vorgesehen ist.

18. Das Instrument gemäß Anspruch 1, bei dem das Signalanalysemodul (SAM, **33**) eine Kombination einer Autokorrelation und eines Nulldurchgangs oder einer spitzenbasierten Tonhöheerfassung implementiert.

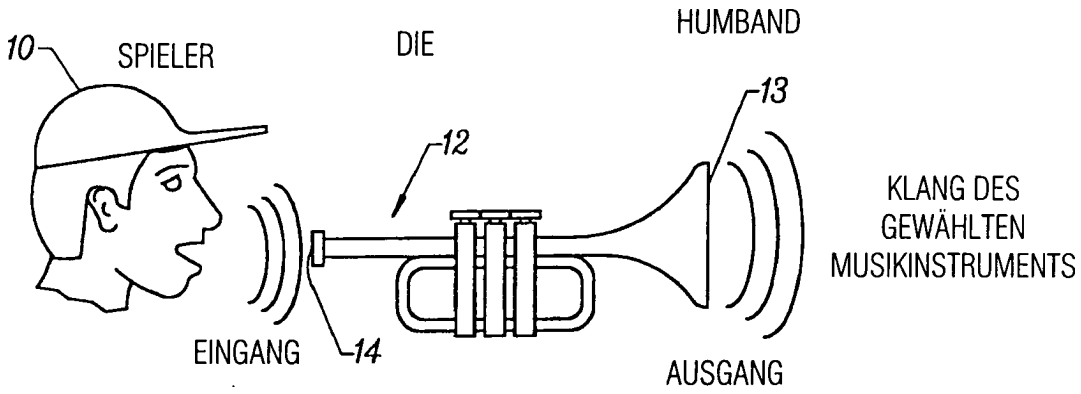
19. Das Instrument gemäß Anspruch 6, bei dem die eine oder die mehreren Benutzersteuerungen (FAC, **37**, **1a**, **1b**) ferner eine beliebige der folgenden umfasst beziehungsweise umfassen:
eine Musikmodus-Auswahltaste und eine Tonika-Einstelltaste für einen Musikmodus, wobei die Tonikaeinstellung eine Auswahl dessen umfasst, ob eine aktuelle Tonhöhe ein erster Ton in dem Musikmodus ist.

20. Das Instrument gemäß Anspruch 6, bei dem die eine oder die mehreren Benutzersteuerungen (FAC, **37**, **1a**, **1b**) ferner folgendes Merkmal umfasst beziehungsweise umfassen:
zumindest eine Instrumentenauswahltaste (FAC, **37**, **1a**, **1b**) zum Bewirken eines dauerhaften oder vorübergehenden Instrumentenwechsels, optional durch dauerhaftes oder vorübergehendes Zuweisen eines Instruments zu einer Taste (FAC, **37**, **1a**, **1b**), wobei ein Pressen der Taste (FAC, **37**, **1a**, **1b**) einen durch das Instrument erzeugten Klang zu einem Klang eines anderen Instruments verändert, das der Taste (FAC, **37**, **1a**, **1b**) zugewiesen ist, bis die Taste (FAC, **37**, **1a**, **1b**) losgelassen oder gewechselt wird.

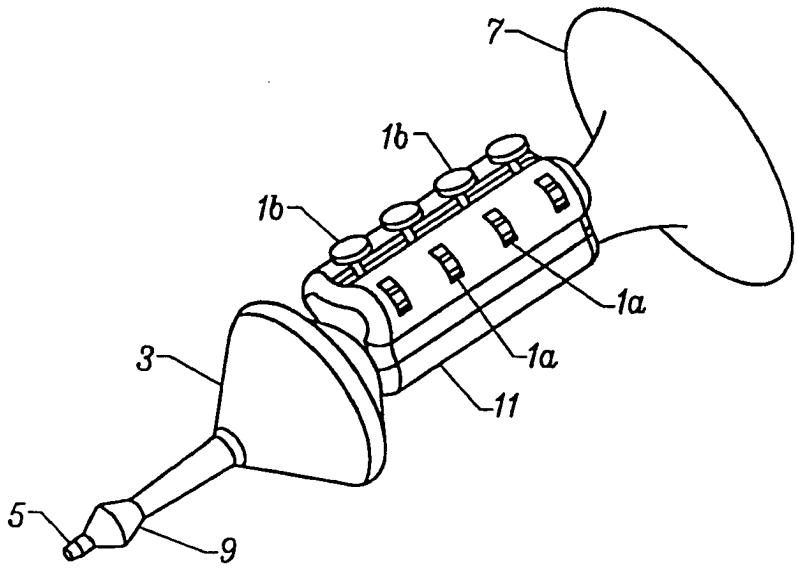
21. Das Instrument gemäß Anspruch 13, bei dem das Gehäuse (**11**) folgende Merkmale aufweist:
einen Innenbehälter und einen Außenmantel;
wobei die Stromquelle in dem Außenmantel enthalten ist;
wobei der Innenbehälter das Signalanalysemodul (SAM, **33**) enthält; und
wobei der Außenmantel so hergestellt ist, dass er einem traditionellen Musikinstrument ähnelt.

Es folgen 29 Blatt Zeichnungen

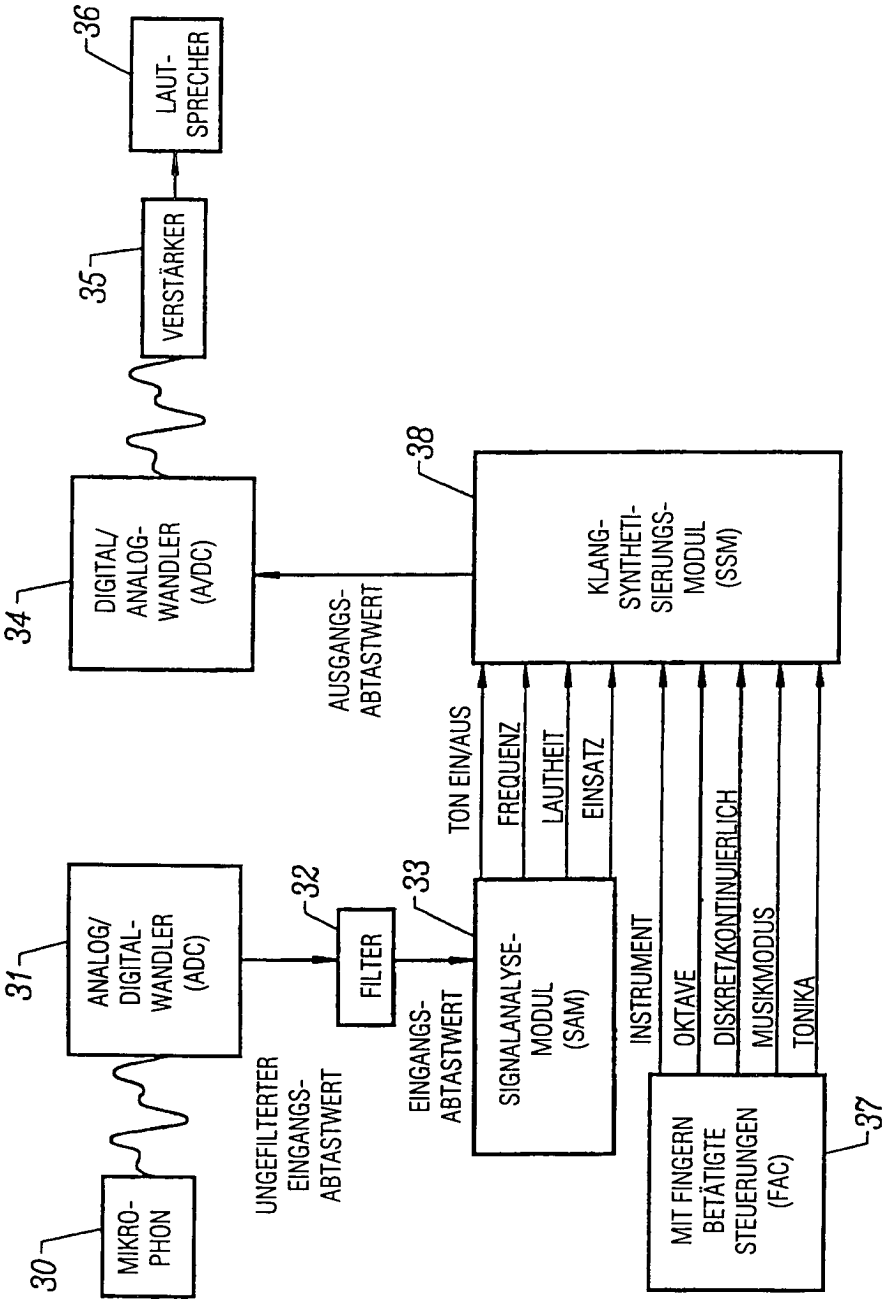
Anhängende Zeichnungen



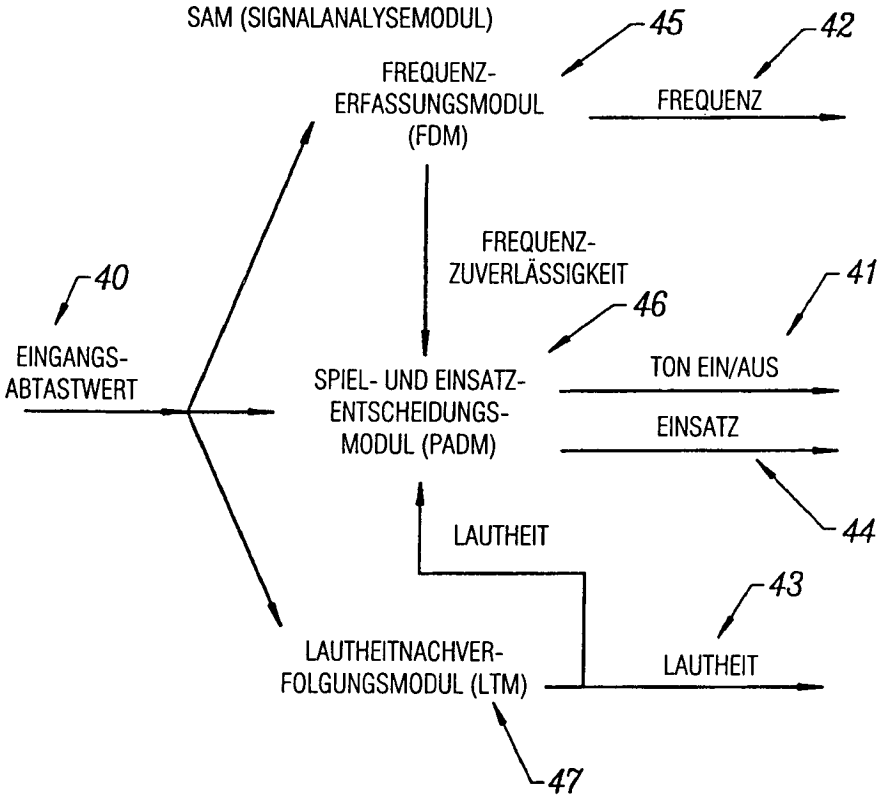
FIGUR 1



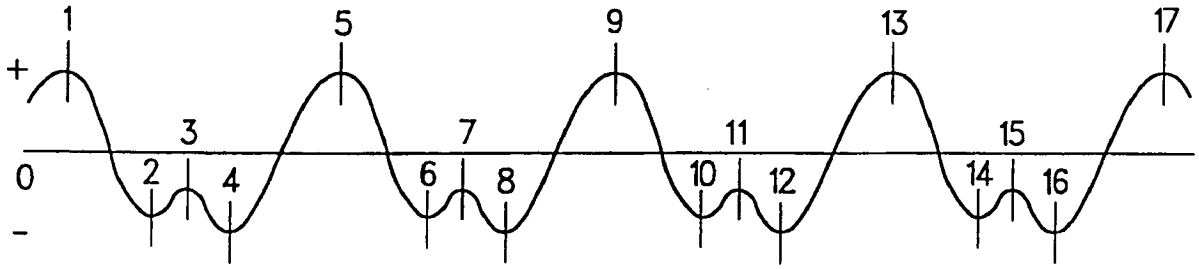
FIGUR 2


30
MIKRO-
PHON
31
ANALOG/
DIGITAL-
WANDLER
(ADC)
UNGEFILTERTER
EINGANGS-
ABTASTWERT
32
FILTER
33
EINGANGS-
ABTASTWERT
SIGNALANALYSE-
MODUL
(SAM)
TON EIN/AUS
FREQUENZ
LAUTHEIT
EINSATZ
INSTRUMENT
OKTAVE
DISKRET/KONTINUIERLICH
MUSIKMODUS
TONIKA
MIT FINGERN
BETÄTIGTE
STEUERUNGEN
(FAC)
37
38
KLANG-
SYNTHETI-
SIERUNGS-
MODUL
(SSM)
AUSGANGS-
ABTASTWERT
34
DIGITAL/
ANALOG-
WANDLER
(A/DC)
35
VERSTÄRKER
36
LAUT-
SPRECHER

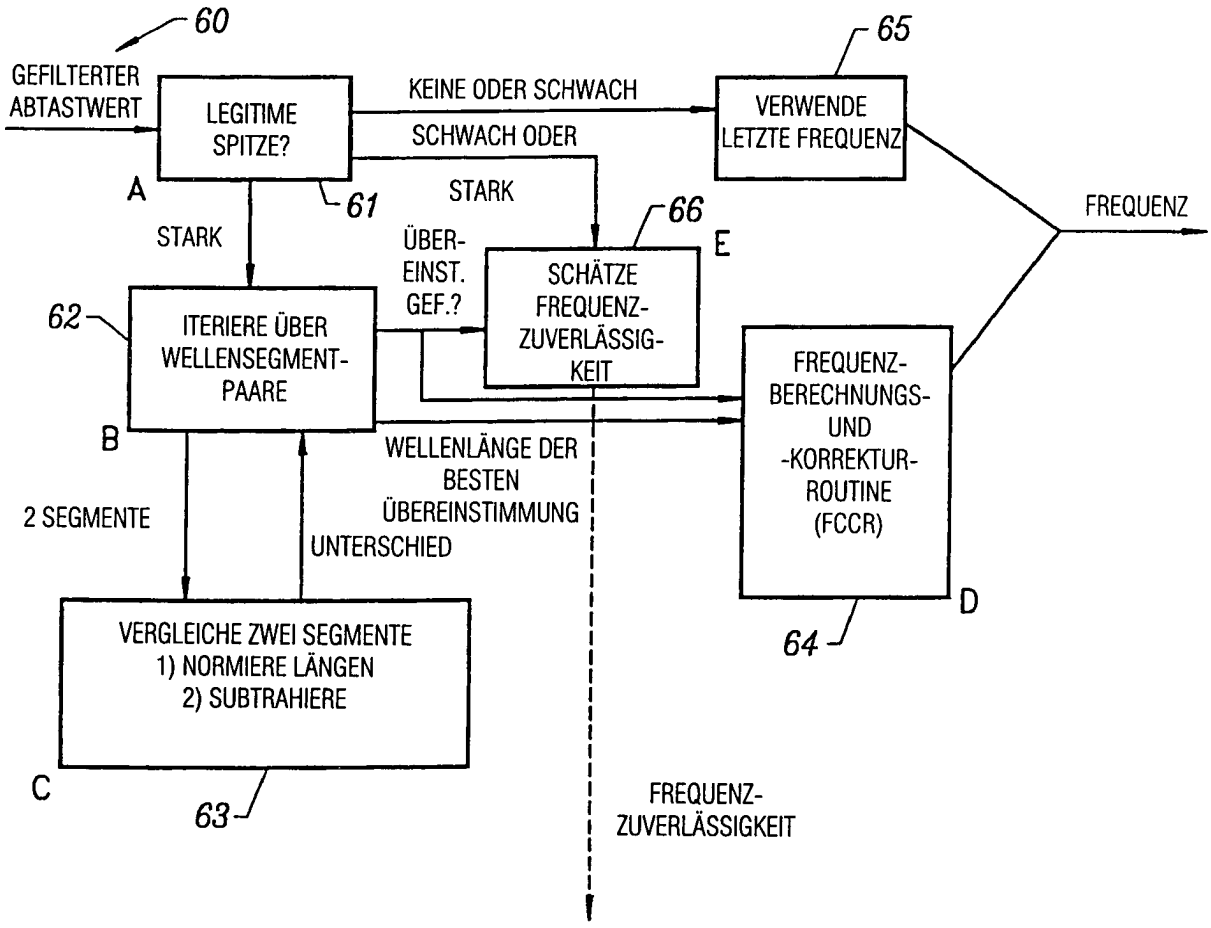
FIGUR 3



FIGUR 4



FIGUR 5


60
GEFILTERTER ABTASTWERT
LEGITIME SPITZE?
A
61
KEINE ODER SCHWACH
SCHWACH ODER STARK
65
VERWENDE LETZTE FREQUENZ
FREQUENZ
STARK
ÜBER-EINST. GEF.?
66
E
SCHÄTZE FREQUENZ-ZUVERLÄSSIG-KEIT
62
ITERIERE ÜBER WELLENSEGMENT-PAARE
B
FREQUENZ-BERECHNUNGS- UND -KORREKTUR-ROUTINE (FCCR)
D
64
WELLENLÄNGE DER BESTEN ÜBEREINSTIMMUNG
2 SEGMENTE
UNTERSCHIED
VERGLEICHE ZWEI SEGMENTE
1) NORMIERE LÄNGEN
2) SUBTRAHIERE
C
63
FREQUENZ-ZUVERLÄSSIGKEIT

FIGUR 6

```
1  Define:  t ② the current time step, t
2           sample (t)② the sample value at t
3           slope (x)② sample (x) - sample (x-1)
4  BEGIN
5  high_peak ②TRUE if and only if: slope(t) < 0 AND slope(t-1) > 0
6  low_peak  ②TRUE if and only if: slope(t) > 0 AND slope(t-1) < 0
7  curvature ② |slope(t) - slope(t-1)|
8  IF NOT high_peak AND NOT low_peak
9     peak②NONE
10 ELSE IF  |sample|②< MIN_SAMPLE_FOR_PEAK
11    peak②weak
12 ELSE IF curvature < MIN_CURVATURE_FOR_PEAK
13    peak②weak
14 ELSE
15    peak② STRONG
16 END
```

FIGUR 7

```
1       Define:  wave1 ② the wave segment from start to split:  wave(start, split)
2                wave2 ② the wave segment from split to current:  wave(split, current)
3                length(wave(x,y)) ②  t_y − t_x
4                sample(peak) ② the sample value in the wave corresponding to current
5       BEGIN
6       match_found ② FALSE
7       FOR ALL (start, split) SUCH THAT
8       (MIN_WAVELENGTH ≤ length(wave1) < MAX_WAVELENGTH)
9   AND (MIN_WAVELENGTH ≤ length(wave2) < MAX_WAVELENGTH)
10  AND | length(wave1) − length(wave2) | / length(wave1) ≤ MAX_LENGTH_RATIO
11  AND | sample(start) − sample(split) | ≤ MAX_AMPLITUDE_DIFFERENCE
12  AND | sample(split) − sample(current) | ≤ MAX_AMPLITUDE_DIFFERENCE)
    {
13     difference ② compare(wave1, wave2)
14     IF (note_on AND (difference < MAX_DIFFERENCE_WHEN_NOTE_ON))
15        OR (NOT note_on AND (difference < MAX_DIFFERENCE_WHEN_NOTE_OFF)) {
16           wavelength = (length(wave1) + length(wave2)) / 2
17           IF (wavelength  2 * best_wavelength)
18              EXIT
19           IF (NOT match_found) OR (difference < best_match) {
20              best_match ② difference
21              best_wavelength ② wavelength
22              match_found ② TRUE
             }
        }
    }
23  END
```

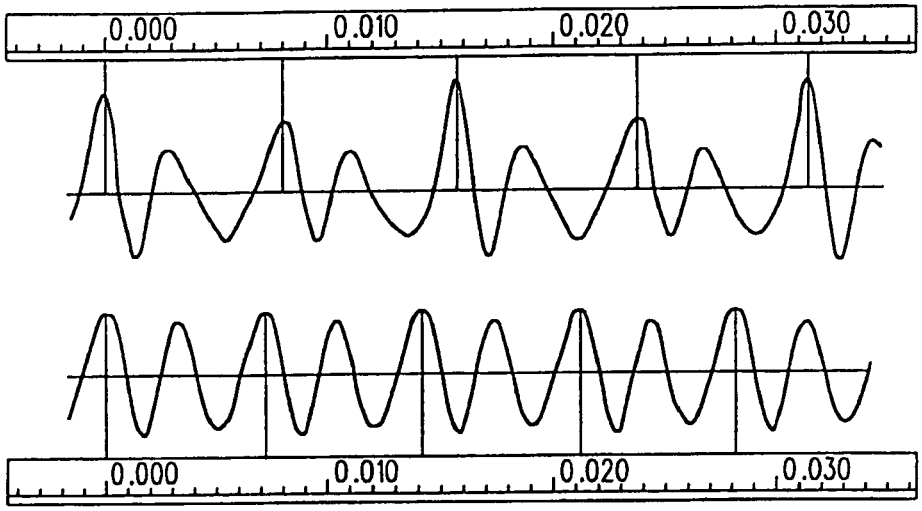
FIGUR 8

```
    Compare(wave1, wave2)

1   BEGIN
2     N_checkpoints ② length(wave1) / N_SAMPLES_PER_CHECKPOINT
3     wavelength_ratio ② length(wave2) / length(wave1)

4     difference ② 0
5     i ② 0
6     WHILE (i < N_checkpoints) {
7         t1 ② t_start + i * N_SAMPLES_PER_CHECKPOINT
8         t2 ② t_split + i * N_SAMPLES_PER_CHECKPOINT * wavelength_ratio

9         s1 ② sample(t1)
10        s2 ② sample(t2)
11        difference ② difference + | s1 - s2 |
12        mean_magnitude ② ( | s1 | + | s2 | ) / 2
13        IF (mean_magnitude > mean_magnitude_max)
14        mean_magnitude_max ② mean_magnitude

15        i ② i + i
      }
16    difference ② difference / (N_checkpoints * mean_magnitude_max)
17    return(difference)
18  END
```

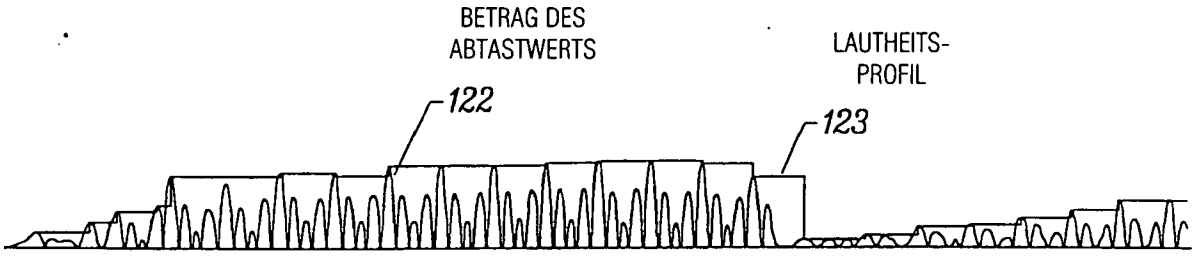
FIGUR 9

```
    FREQUENZBERECHNUNGS- UND -KORREKTURROUTINE

1   BEGIN

2   time_span ② t - time_of_last_match
3   suggested_frequency ② 1 / wavelength

4   IF suggested_frequency > last frequency THEN
5       frequency_change ② (suggested_frequency / last frequency)-1
6   ELSE
7       frequency_change ② (last frequency / suggested_frequency)-1
8   IF no match was found
9       frequency ② last frequency
10  ELSE {
11      time_of_last_match ② t
12      IF (frequency_change / time_span) < MAX_VOICE_SPEED THEN
13          frequency ② suggested_frequency
14      ELSE
15          frequency ② last frequency

    }
16  END
```

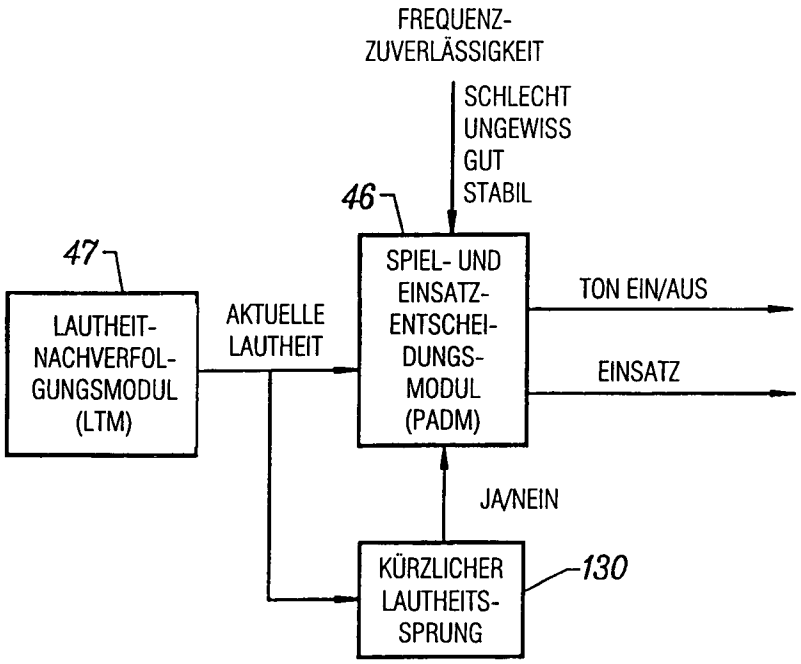
FIGUR 10



FIGUR 11



FIGUR 12



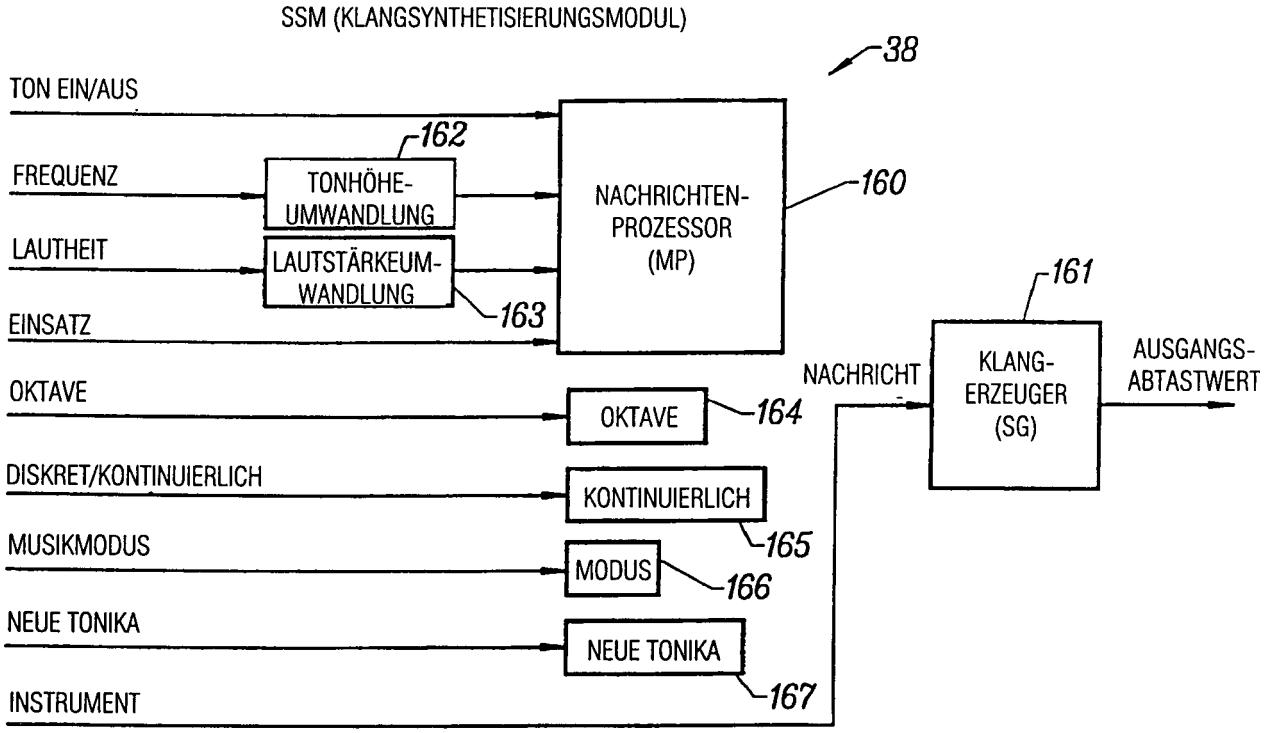
FIGUR 13

```
    SPIEL- UND EINSATZENTSCHEIDUNGSMODUL
1   Define:  t - the current time
2   BEGIN
3   IF    ( frequency reliability is GOOD or STABLE
4         AND  there has been a recent loudness surge
5         AND  (t - time_of_last_attack) < MIN_ATTACK_GAP
          )
7     OR ( frequency reliability is STABLE
8         AND current loudness > MIN_ATTACK_LOUDNESS
9         AND note is OFF
10        AND (t - time_of_play_off) > MIN_TIME_OFF
          )
11  THEN {
12      note                    ② ON
13      attack                  ② TRUE
14      loudness surge          ② FALSE
15      time_of_last_attack     ② t
    }
16  ELSE {
17      attack ② FALSE
18      IF (frequency reliability is BAD) OR (loudness < MIN_LOUDNESS)
    THEN {
19          note                ② OFF
20          time_of_play_off    ② t
        }
    }
21  END
```

FIGUR 14

```
SCHÄTZE FREQUENZZUVERLÄSSIGKEIT

BEGIN
  IF peak quality is WEAK THEN {
     consecutive_weak ② consecutive_weak + 1
     IF (consecutive_weak > MAX_CONSECUTIVE_WEAK)  THEN {
        bad_frequency_signs ② bad_frequency signs + 1
        good_frequency_signs ② 0
        consecutive_weak ② 0
     }
  }
  ELSE IF no frequency match was found THEN {
        bad_frequency_signs ② bad_frequency signs + 1
        good_frequency_signs ② 0
        consecutive_weak ② 0
  }
  ELSE {
    // frequency match was found
    good_frequency_signs ② good_frequency_signs + 1
    bad_frequency_signs ② 0
    consecutive_weak ② 0
  }
  IF (bad_frequency_signs > 5)  THEN
     frequency reliability is BAD
  ELSE IF (good_frequency_signs > 3)  THEN
     frequency reliability is STABLE
  ELSE IF (peak quality is STRONG AND a frequency match was found) THEN
     frequency reliability is GOOD
  ELSE
     frequency reliability is UNSURE

END
```

FIGUR 15


SSM (KLANGSYNTHETISIERUNGSMODUL)
38
TON EIN/AUS
FREQUENZ
LAUTHEIT
EINSATZ
OKTAVE
DISKRET/KONTINUIERLICH
MUSIKMODUS
NEUE TONIKA
INSTRUMENT
162
TONHÖHE-UMWANDLUNG
LAUTSTÄRKEUM-WANDLUNG
163
NACHRICHTEN-PROZESSOR (MP)
160
161
NACHRICHT
KLANG-ERZEUGER (SG)
AUSGANGS-ABTASTWERT
OKTAVE
164
KONTINUIERLICH
165
MODUS
166
NEUE TONIKA
167

FIGUR 16

```
1   BEGIN

2   pitch ② pitch − synthesizer_offset + 12 * octave
3   IF continuous THEN
4     mode ② chromatic
5   nearest_note ② nearest_mode_note(pitch, mode, new_tonic)

6   IF NOT attack AND note_on THEN
7      pitch_difference ② pitch − current_note
8      IF continuous THEN
9         IF| pitch_difference | > MAX_BEND_RANGE
10             attack ② TRUE
       }
11     ELSE IF (nearest_note ? current_note) THEN
12        IF | pitch_difference | > |nearest_note − current_note| *
   MAX_PITCH_ERROR
13             attack ② TRUE
14  IF attack THEN
15     current_volume ②volume
16     current_note ② nearest_note
17     IF SG is currently playing THEN
18        send SG message: stop play
19     send SG message: play current_note at current_volume

20     IF continuous THEN
21        adjust_pitch (pitch − current_note)

22     time_of_last_change ② t
       }
23  ELSE IF SG is currently playing THEN {
24     IF note is OFF THEN {
25        send SG message: stop play
       }
26     ELSE IF (t − time_of_last_change > SG_REFRACTORY_PERIOD) THEN {
27        IF NOT (volume  current_volume) THEN {
28           current_volume ② volume
29           send SG message: set volume to current_volume
30           time_of_last_change ② t
          }
31        IF (continuous AND pitch_difference ? 0) THEN {
32           adjust_pitch(current_note, pitch_difference)
33           time_of_last_change ② t
          }

       }
34  END
```

FIGUR 17

```
    nearest_mode_note(pitch, mode, new_tonic)
1   Define: Major:      {0,2,4,5,7,9,11,12}
2           Minor:      {0,2,3,5,7,8,11,12}
3           Blues:      {0,3,5,6,7,10,12}
4           Chromatic: {0,1,2,3,4,5,6,7,8,9,10,11,12}
5   BEGIN
6   IF new_tonic THEN
7       tonic    round(pitch) modulo 12
8   offset       (pitch - tonic + 12) modulo 12
9   mode_note    value in specified mode nearest to offset
10  correction   mode_note - offset
11  return(pitch + correction)
13  END
```

FIGUR 18

```
    CFindPitch::CFindPitch()
    {
1       int i;
2       I_peak=0;
3       I_buffer=0;
4       I_tic=0;
5       Play_note=0;
6       Loudness=0;
7       T_tic=0;
8       Freq=0;
9       N_bad_freqs=0;
10
11      Curvature_threshhold=((double) CURVE_THRESHHOLD) *22000.0/((double) SAMPLE_RATE);
12
13      Max_freq_decay_factor=T_DEL_OCTAVE_REDUCTION_MIN*double(SAMPLE_RATE);
14      Max_freq_decay_factor=pow(2.0-1.0/Max_freq_decay_factor);
15
16      int n_code_points_per_highest_freq_wave=5;//# of code points for 360 deg of highest freq.
17
18      T_peak=    new int[PEAK_FEATURE_BUFFER_SIZE];
19      Goodness_peak=  new int[PEAK_FEATURE_BUFFER_SIZE];
20      Ampl_peak=   new double[PEAK_FEATURE_BUFFER_SIZE];
21      Ampl_peak_mag=   new double[PEAK_FEATURE_BUFFER_SIZE];
22      Freq_peak_buffer=new double[PEAK_FEATURE_BUFFER_SIZE];
23
24      T_loudness_tic = new int[PEAK_FEATURE_BUFFER_SIZE];
25      T_peak_long_time_counter = new int[PEAK_FEATURE_BUFFER_SIZE];
26      Loudness_time_buffer = new double[PEAK_FEATURE_BUFFER_SIZE];
27
28      for(i=0; i < PEAK_FEATURE_BUFFER_SIZE; i++){
29            Ampl_peak[i] = Ampl_peak_mag[i] = 0;
30            T_peak[i] = 0
31            Goodness_peak[i] = -1;
32            Freq_peak_buffer[i] = 0;
33            T_loudness_tic[i] = 0;
34            T_peak_long_time_counter[i] = 0;
35            Loudness_time_buffer[i] = 0;
36      }
```

FIGUR 19

```
37
38    int freq_lowest_expected = 50;
39    int freq_highest_expected = 400;
40
41    Time_ratio_error_consecutive_wavelengths_reject = 0.12;
42    N_points_raw_sound_buffer = (2*SAMPLE_RATE)(freq_lowest_expected;
43    N_samples_longest_expected_wavelength - int(1.05*double(SAMPLE_RATE)/ double(freq_lowest_expected));
44    int n_samples_per_wave_length_hi_freq = int(double(SAMPLE_RATE)/double (freq_highest_expected));
45    N_samples_shortest_expected_wavelength=int(0.97* double(n_sample_per_wave_length_hi_freq));
46
47    int n_code_points_per_wave_lo_freq=
48          n_code_points_per_highest_freq_wave*freq_highest_expected/freq_lowest_ expected + 20;
49    N_samples_per_code_point=
50          n_samples_per_wave_length_hi_freq/n_code_points_per_highest_freq_wave;
51
52    Ampl_buffer-new double[N_points_raw_sound_buffer];
53    for(i=0; i < N_points_raw_sound_buffer;i++)Ampl_buffer[i] = 0
54
55    Code_vector=  new double[n_code_points_per_wave_lo_freq];
56    Code_vector_previous = new double[n_code_points_per_wave_lo_freq];
57    Freq_half_note_divider = new double[50];
58
59    int  n_order = 3;
60    int  n_passes = 1;
61    double freq_low = 200;
62    double freq_high = 300; C62
63
64    Hof = new DHighOrdFilt();
65    Hof->Initialize(n_order, n_passes, freq_low, freq_high);
66    Filter_scale_factor = 2.0*pow(3.5, ((double) n_order));
  }
```

FIGUR 20

```
void CfindPitch::GetPitch(double &freq, double &loudness, double ampl_raw,
            double &ampl_filt, int &got_new_freq, int &play_note,
                    int &got_note_attack, int &note)
        {
    1       int i,j;
    2       int i_peak_last, i_peak_last_last;
    3       int t_del, t_del_last, t_del_prev;
    4       Double rating, freq_orig;
    5       Static int note_last=0;
    6       Static double freq_last=0;
    7       Static double slope_last=0;
    8       int corrected_the_freq=0;
    9       Static int n_steps_skip_loudness_buf=0;
    10      Freq=Freq;
    11      Play_note = Play_note;
    12      Loudness = Loudness;
    13      t_del_prev = 1000000;
    14      Rating_min - 100.0;
    15      int wavelength_best=0;
    16      int t_del_prev_match=0, t_del_last_prev_match=0;
    17      int n_potential_matches = 0;
    18
    19      Ampl_filt=HOF->Filter(ampl_raw);
    20      Got_note_attack = 0;//for record keeping only
    21
    22      int i_buffer_last = I_buffer;
    23      T_tic = Mod_time_counter(T_tic + 1);
    24      I_buffer = Mod_sound_buffer(I_buffer + 1);
    25      Ampl_buffer[I_buffer]=ampl_filt;
    26
    27      Double slope = ampl_filt - Ampl_buffer[I_buffer_last];
    28
    29      int got_peak = (slope < 0) && (slope_last > 0);
    30      int got_valley = (slope > 0)&&(slope_last < 0);
    31      int got_extremem = got_valley || got_peak;
    32
    33      Double curvature = fabs(slope - slope_last);
    34
    35      Slope_last = slope;
    36
    37      int got_strong_extremem = got_extremem&&(ampl_filt > 10.0)
    38                                          &&(curvature > CURVE_THRESHOLD);
    39      if(!got_extremem){
    40            goto out_for;
    41      }
```

FIGUR 21

GetPitch() continued □.

```
42  //--------------we are at a PEAK or Valley----------------------
43  got_new_freq=1;
44  i_peak_last=I_peak;
45  I_peak=Mod_peaks(I_peak + 1);
46  T_peak[I_peak] = I_buffer;
47  T_peak_long_time_counter[I_peak] = T_tic;
48  Ampl_peak[I_peak] = ampl_filt;
49  Goodness_peak[I_peak] = -1;
50  Freq = freq = Freq_peak_buffer[I_peak] = Freq_peak_buffer[i_peak_last];
51  T_tic_last_good_pitch = T_tic;
52
53
54  if(!got_strong_extremum){
55      Goodness_peak[I_peak] = 0
56      goto out_far;
57  }
58  got_new_freq = 1;
59
60  for(i = -1; i>-15; i--){
61      i_peak_last - Mod_peaks(I_peak + 1);
62      t_del=Mod_sound_buffer(T_peak[I_peak]-T_peak[i_peak_last]);
63
64      if(t_del>N_samples_longest_expected_wavelength){
65          got out_near; //peak too far away: we couldn't get a match
66      }
67      else if(t_del<N_samples_shortest_expected_wavelength){
68          continue; //peak too far away: we couldn't get a match
69      }
70      for(j=i-1; j>i-15; j--){
71          i_peak_last_last=Mod_peaks(I_peak+j);
72          t_del_last=Mod_sound_buffer(T_peak[i_peak_last]-T_peak[i_peak_last_last] )
73
74          if(t_del_last>N_samples_longest_expected_wavelength){
75                  break;
76          }
77          if(t_del_last>N_samples_shortest_expected_wavelength){
78                  continue;
79          }
80          rating = Code_match(I_peak, i_peak_last, i-Peak_last_last);
81
82          if((!Play_note&& rating > RATE_LIMIT_LO)
83                                  ||(Play_note&& rating > RATE_LIMIT_HI)){
84                  continue;
85          }
```

FIGUR 22

GetPitch() continued □.

```
86              n_potential_matches++;
87              if(n_potential_matches==1){
88                   Rating_min = rating;
89                   wavelength_best=t_del;
90                   t_del_prev_match = t_del;
91                   t_del_last_prev_match = t_del_last;
92                   if(rating < 0.50){
93                       goto out_near;
94                   }
95                   continue;
96              }
97              double time_diff_ratio =
98               (double) abs(t_del_last-t_del_prev_match-t_del_last_prev_match);
99
100             time_diff_ratio/=(double) t_del_last;
101
102             if(time_diff_ratio < 0.10){
103                  goto out_near;
104             }
105             if(rating > Rating_min)
106                continue;
107             Rating_min = rating;
108             wavelength_best = t_del;
109             t_del_prev_match = t_del;
110             t_del_last_prev_match = t_del_last;
111             if(rating < 0.05){
112                  goto out_near;
113             }
114
115         }
116     }
117 out_near:
118     if(n_potential_matches==0){
119         goto out_far;
120     }
121     Freq = freq = double(SAMPLE_RATE)/double(wavelength_best);
122     Goodness_peak[l_peak] = 1;
```

FIGUR 23

```
                                        GetPitch() continued□.
122
123         Freq_orig = freq;
124         if(Correct_freq_doubling_or_halving_using_local_past(freq)){
125             corrected_the_freq = 1;
126         }
127         Freq = Freq_peak_buffer[l_peak] = freq;
128
129         N_peaks_since_got_good_freq = 0;
130         if(CONVERT_TO_HALF_TONES){
131             freq = find_nearest_half_note(freq, note);
132         }
133
134 out_far:
135         Static int got_note_decay;
136         if(got_extremum){
137             Loudness = loudness = Get_loudness(ampl_filt, freq);
138         }
139         If(++n_steps_skip_loudness_buf==N_STEPS_SKIP_LOUDNESS_BUF){
140             l_tic = Mod_loudness_tics(l_tic + 1);
141             Loudness_time_buffer[l_tic] = Loudness;
142             T_loudness_tic[l_tic] = T_tic;
143             n_steps_skip_loudness_buf = 0;
144         }
145         if(got_extremum){
146             int got_note_attack-with_new_freq;
147             Get_attack_and_play_condition(got_note_attack, got_strong_extremum,
148                                                 got_note_attack_with_new_freq);
149             if(corrected_the_freq){
150                 if(got_note_attack_with_new_freq){
151                     Freq_peak_buffer[l_peak] = Freq = freq_orig;
152                 }
153             }
154         }
155         play_note = Play_note;
    }
```

FIGUR 24

```
double CFindPitch::Code_match(int i_peak, int i_peak_last, int i_peak_last_last)
{
1    if(fabs(Ampl_peak[i_peak]-Ampl_peak[i_peak_last])
2      >MAX_EXPECTED_CONSECUTIVE_PEAK_HEIGHT_DIFFERENCE){
3          return 100;
4
5    if(fabs(Ampl_peak[i_peak]-Ampl_peak[i_peak_last_last])
6      >MAX_EXPECTED_CONSECUTIVE_PEAK_HEIGHT_DIFFERENCE){
7          return 100;
8    }
9    int_peak = T_peak[i_peak];
10   int t_peak_last  =T_peak[i_peak_last];
11   int t_peak_last_last = T_peak[i_peak_last_last];
12
13   int t_del=  Mod_sound_buffer(t_peak  -t_peak_last);
14   int t_del_last = Mod_sound_buffer(t_peak_last-t_peak_last_last);
15
16   double time_del_ratio-fabs(((double)(t_del-t_del_last))/((double)t_del)) ;
17
18   if(fabs(time_del_ratio)>Time_ratio_error_consecutive_wavelengths_reject) ;
19       return 200;
20   }
21   int i;
22   int n_code_points = t_del/N_samples_per_code_point;
23   double n_samples_per_code_point
24            = double(t_del)/double(n_code_points);
25   double n_samples_per_code_point_last
26            = double(t_del_last)/double(n_code_points);
27
```

FIGUR 25

Code_match() continued □

```
27
28    double dist;
29    double time_back=0, time_back_last=0;
30    int time  =t_peak;
31    int time_last=t_peak_last;
32
33    double distance = 0.0;
34    double ampl=0, ampl_max=-100.0;
35
36    for(i=0; i<n_code_points; i++){
37         Dist = fabs(Ampl_buffer[time] - Ampl_buffer[time_last];
38         Ampl = (fabs(Ampl_buffer[time]) + fabs(Ampl_buffer[time_last]));
39
40         if(ampl>ampl_max)
41              ampl_max = ampl;
42
43         Distance +=dist;
44         Time_back +=n_samples_per_code_point;
45         Time_back_last +=n_samples_per_code_point_last;
46
47         Time  =Mod_sound_buffer(t_peak  -round(time_back));
48         Time_last=Mod_sound_buffer(t_peak_last-round(time_back_last));
49    }
50    Distance=2.0*distance/(double(n_code_points)*ampl_max);
51    return distance;
   }
```

FIGUR 26

```
int CfindPitch::Correct_freq_doubling_or_halving_using_local_past(double & freq)
{
1       int i;
2       int i_peak_prev;
3       double freq_prev;
4       int n_peaks_look_back = 4;
5
6       for(i=1; i<=n_peaks_look_back; i++){
7            i_peak_prev=Mod_peaks(I_peak-1);
8
9       freq_prev = Freq_peak_buffer[i_peak_prev];
10
11      if(freq_prev==0)
12           return 0; //only at the VERY beginning
13
14      if(Goodness_peak[i_peak_prev]<=0)
15           continue;
16      if(fabs(freq_prev-freq) > 40){
17          Freq = freq = freq_prev;
18           return 1;
19      }
20    }
21    return 0;
22  }
```

FIGUR 27

```
double CFindPitch::Get_loudness(double ampl_filt, double freq)
{
1     int t_del;
2     double wave_length;
3     int time_stretch_look_in_the_past;
4     int t_del_for_max_peak=0;
5     int i_peak_prev, i;
6     static double loudness_rate =  0;
7     static double loudness_target =  0;
8     double stretch_factor;
9
10    static int play_note_=0;
11    double ampl_filt_mag = fabs(ampl_filt);
12    if(freq==0){
13         //only at the very beginning
14         time_stretch_look_in_the_past = N_samples_longest_expected_wavelength;
15    }
16    else{
17        wave_length = SAMPLE_RATE/freq;
18        stretch_factor - pow(Max_freq_decay_factor,
19                  double(Mod_time_counter(T_tic-T_tic_last_good_pitch)));
20        time_stretch_look_in_the_past = round(stretch_factor*wave_length);
21        time_stretch_look_in_the_past = imin(N_samples_longest_expected_wavelength,
22                                                time_stretch_look_in_the_past);
23    }
24    //This for-loop finds the largest (plus) peak over the
25    //most recent(time_stretch_look_in_the_past)samples;
26    double ampl_max=ampl_filt_mag, ap;
27    int t_peak_current = T_peak[I_peak];
28
29    for(i=1; i<10; i++){
30         i_peak_prev=Mod_peaks(I_peak-i);
31         t_del=Mod_sound_buffer(t_peak_current-T_peak[i_peak_prev]);
32         if(t_del>time_stretch_look_in_the_past)
33              break;
34         ap=fabs(Ampl_peak[i_peak_prev]);
35         if(ap>ampl_max){
36             ampl_max=ap;
37             t_del_for_max_peak=t_del;
38         }
39    }
40    Return Loudness=ampl_max;
}
```

FIGUR 28

```
int CFindPitch::Check_for_loudness_attack()
{
    static double LOUDNESS_RATIO_ATTACK_ACCEPT = 0.65;
    static int T_DEL_ATTACK_REFRACTION=int(double(SAMPLE_RATE)*0.1); //0.1 secs
    static int N_CONSECUTIVE_LOW_TICS_REQUIRED = 2;
    int i, n_tics_low=0, i_last=0;
    int i_tic_look_back, t_del_from_now_till_I_tic_prev;
    double ampl_min = 100000;

    static int t_loudness_attack_refraction_over=-1;

    double loudness_low_accept=
          LOUDNESS_RATIO_ATTACK_ACCEPT*Loudness_time_buffer[I_tic];
    int t_tic_look_back;

    static int t_tic_last_attack=-1;
    int t_tic_current = T_loudness_tic[I_tic];

    if(t_tic_current<t_loudness_attack_refraction_over){
          Return 0;
    }
    if(Loudness_time_buffer[I_tic]<LOUDNESS_MIN_REQUIRED_FOR_LOUDNESS_ATTACK
          Return 0;

    for(i=1;i<PEAK_FEATURE_BUFFER_SIZE; i++){
         i_tic_look_back = Mod_peaks(I_tic-i);
         t_del_from_now_till_i_tic_prev=
              Mod_time_counter(t_tic_current-T_loudness_tic[i_tic_look_back];

         if(t_del_from_now_till_i_tic_prev>N_STEPS_LOOK_BACK_ATTACK)
               break;

         if(Loudness_time_buffer[i_tic_look_back]<loudness_low_accept){
              if((i-i_last)>1){
                   n_tics_low=1;
                   t_tic_look_back=T_loudness_tic[i_tic_look_back];
                   if(t_tic_look_back==t_tic_last_attack){
                         return 0;
                   }
              }
              else{
                   n_tics_low++;
                   if(n_tics_low>=N_CONSECUTIVE_LOW_TICS_REQUIRED){
                        t_tic_last_attack=t_tic_current;
                        t_loudness_attack_refraction_over=
                             Mod_time_counter(t_tic_current + T_DEL_ATTACK_REFRACTION);
                        return 1;
                   }
              }
              i_last=i;
         }
    }
    return 0;
}
```

FIGUR 29

```
    int CFindPitch::Get_goodness_of_the_frequency()
1   {
2           int i;
3           static int T_DEL_WITH_BAD_FREQS_FOR_NOTE_TURN_OFF = int(double(SAMPLE_RATE)*0.05);
4           static int N_PEAKS_IN_PAST_WITH_GOOD_FREQ_FOR_HALFWAY_GOOD_FREQ=1;
5           static int N_GOOD_PEAKS_IN_PAST_WITH_GOOD_FREQ_TO_HAVE_REALLY_GOOD_FREQ= 3;
6           int goodness_peak_prev;
7           int i_peak_prev, t_del;
8           for(i=0; i<100; i++){
9                   i_peak_prev=MOd_peaks(l_peak-1);
10                  t_del=Mod_time_counter(T_peak_long_time_counter[l_peak]
11                                  -T_peak_long_time_counter[i_peak_prev];
12                  if(t_del>T_DEL_WITH_BAD_FREQS_FOR_NOTE_TURN_OFF)
13                        return-1;
14                  if(Goodness_peak[i_peak_prev]>0)
15                        break;
16          }
17          if(Goodness_peak[l_peak]<0)
18                return 0;
19
20          int n_consecutive_bad_curvature_excess=0;
21          if(Freq_peak_buffer[l_peak]<150)
22                 n_consecutive_bad_curvature_excess=7;
23          else if(Freq_peak_buffer[l_peak]<200)
24                 n_consecutive_bad_curvature_excess=6;
25          else if(Freq_peak_buffer[l_peak]<250)
26                 n_consecutive_bad_curvature_excess=5;
27
28          int n_consecutive_bad_curvature=0, n_good=0;
29          const static int T_DEL_MAX_LOOK_BACK_FOR_VERY_GOOD_FREQ
30                                  =int(double(SAMPLE_RATE)*0.2);//0.2 seconds
31
32          for(i=0; i<100; i++){
33                  i_peak_prev=Mod_peaks(l_peak-i);
34                  t_del=Mod_time_counter(T_peak_long_time_counter[l_peak]
35                          -T_peak_long_time_counter[i_peak_prev]);
36                  if(t_Del>T_DEL_MAX_LOOK_BACK_FOR_VERY_GOOD_FREQ)
37                         return 0;
```

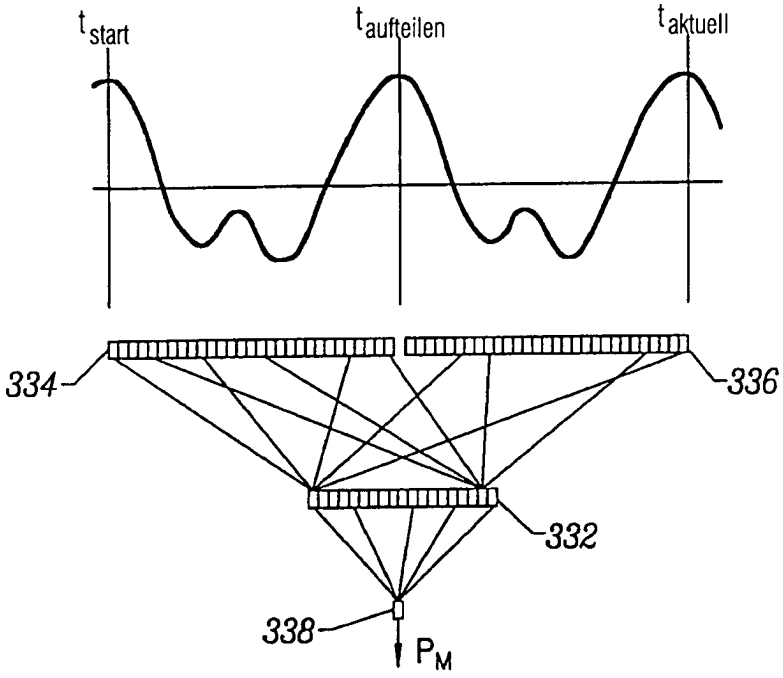
FIGUR 30

```
38          Get_goodness_of_the_frequency() continued □ □

39          goodness_peak_prev = Goodness_peak[i_peak_prev];
40
41          if(     goodness_peak_prev==0){
42                  if(++n_consecutive_bad_curvature>n_consecutive_bad_curvature_excess)
43                          break;
44          }
45          else if(goodness_peak_prev==1){
46                  if(++n_good==
                        N_GOOD_PEAKS_WITH_GOOD_FREQ_TO_HAVE_REALLY_GOOD_FREQ)
47                          return 2;
48                  n_consecutive_bad_curvature=0;
49          }
50          else break;
51      }
52      if(Goodness_peak[I_peak]==1)
53              return 1;
54      return 0;
    }
```

FIGUR 31

```
void CFindPitch::Get_attack_and_play_condition(int&got_attack,
                                               int got_strong_extremum,
                                               int&got_note_attack_with_new_freq)
{
1    static int i_count_down_refraction = 0;
2    static int got_loudness_decay = 0;
3    static int freq_goodness =-1;
4    int got_loudness_attack = 0;
5    const static int TIME_SINCE_PLAY_OFF_MIN_FOR_VERY_GOOD_FREQ_ATTACK
6                          =int(double(SAMPLE_RATE)*0.05);
7    const static double LOUDNESS_THRESHHOLD_FOR_VERY_GOOD_FREQ_ATTACK =;10.0
8
9    static int time_of_play_off=-1;
10   int time_since_play_off;
11   got_note_attack_with_new_freq =got_note_attack = 0
12
13   int freq_goodness_prev=freq_goodness;
14   freq_goodness=Get_goodness_of_the_frequency();
15
16   if(got_strong_extremum){
17        got_loudness_attack = 0
18        if(freq_goodness>0){
19             got_loudness_attack=Check_for_loudness_attack();
20             if(got_loudness_attack){
21                  got_note_attack=1;
22                  Play_note=1;
23                  got_loudness_decay=0;
24                  if(freq_goodness_prev<=0)//note this
25                       got_note_attack_with_new_freq=1;
26             }
27        }
28   }
29   if(!got_note_attack&&!Play_note&&freq_goodness>1{
30
31        time_since_play_off=Mod_time_counter(T_tic-time_of_play_off);
32
33        if(time_since_play_off>TIME_SINCE_PLAY_OFF_MIN_FOR_VERY_GOOD_FREQ_ATTACK){
34             if(Loudness>LOUDNESS_THRESHHOLD_FOR_VERY_GOOD_FREQ_ATTACK){
35                  if(PRINT_OVERVIEW)
36                       fprintf(opo,"Got attack(very good freq)!!\m"
37                  got_note_attack=1;
38                  Play_note=1;
39                  got_loudness_decay=0;
40             }
41        }
42   }
43   if(Loudness<VOLUME_THRESHHOLD||freq_goodness<0){
44        Play_note=0;
45        freq_goodness=-1;
46        time_of_play_off=T_tic;
47   }
}
```

FIGUR 32



FIGUR 33