HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Sebastian Hillinger

Voruntersuchung zur Softwarerealisierung eines Stimmgerätes für Orgelpfeifen

2014

Fakultät **Elektro- und Informationstechnik**

BACHELORARBEIT

Voruntersuchung zur Softwarerealisierung eines Stimmgerätes für Orgelpfeifen

Autor:
Sebastian Hillinger

Studiengang:
Informationstechnik

Seminargruppe:
IT10wK-B

Erstprüfer:
Prof. Dr.-Ing. Thomas Beierlein

Zweitprüfer:
Dr. phil. Markus Voigt

Mittweida, 2014

Bibliografische Angaben

Hillinger, Sebastian: Voruntersuchung zur Softwarerealisierung eines Stimmgerätes für Orgelpfeifen, 49 Seiten, 20 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Elektro- und Informationstechnik

Bachelorarbeit, 2014

Referat

Im Rahmen dieser Arbeit werden Algorithmen zur Messung der Grundtonfrequenz von diskreten Signalen auf ihre Tauglichkeit zur Bestimmung der Tonhöhe von Orgelpfeifen untersucht. Dabei werden Genauigkeit und Störsicherheit unter dem Aspekt der Implementierung auf einem Embedded Linux System betrachtet.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AKF	Autokorrelationsfunktion
CAS	Client Aktor Steuerung
DFT	Diskrete Fourier-Transformation
FFT	Fast Fourier-Transformation
HMT	Hermode Tuning
MIDI	Musical Instrument Digital Interface
NQDF	Normierte quadratische Differenzfunktion
PCM	Pulse Code Modulation
ZAS	Zentrale Aktor Steuerung

1 Einleitung

Im Rahmen des Forschungsprojektes *Pfeifenorgel mit dynamischer Stimmung* wurde ein Instrument entwickelt, welches seine Stimmung während des Spielens selbsttätig, das heißt ohne Zutun des Spielers, nur anhand der gespielten Noten, anpasst. Auf diese Weise wurde eine dynamische Stimmung ermöglicht, die deutlich reinere Intervalle aufwies als die bisher verwendete *gleichstufige Stimmung*. Dies war bisher nur mit elektronischen Instrumenten möglich.

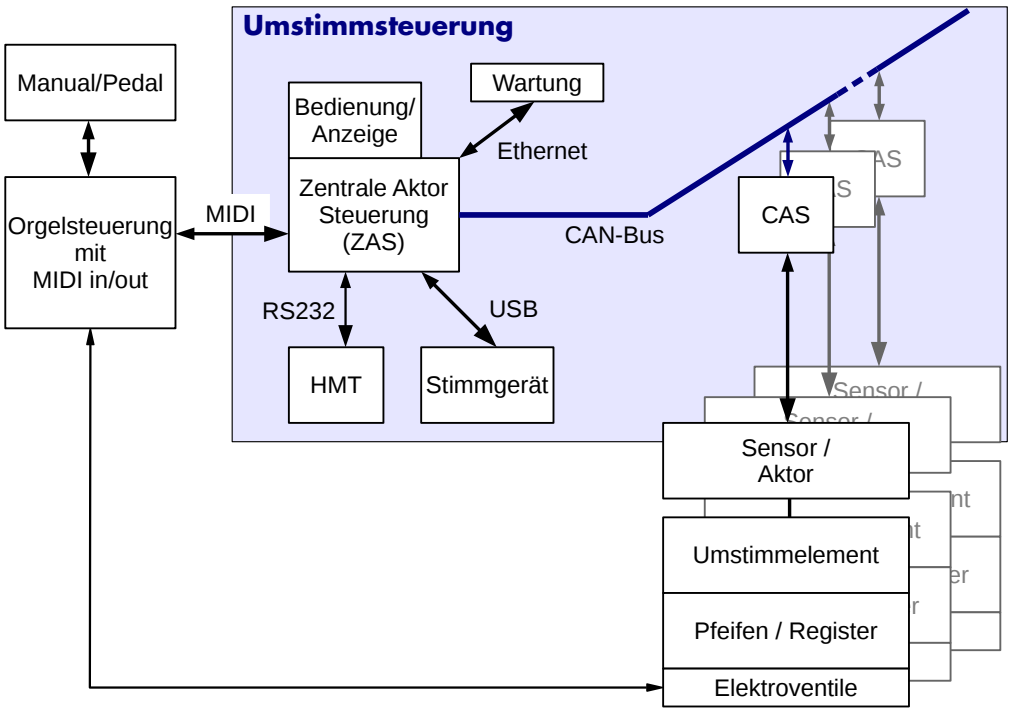


Abbildung 1.1: Blockschaltbild des Gesamtsystems der Pfeifenorgel mit Umstimmsystem [1, S. 7]

Damit das Instrument automatisch umgestimmt werden kann, verfügt jede Pfeife über eine Umstimmeinheit, die mit einem Linearmotor bewegt wird. Die Tonhöhe verändert sich in Abhängigkeit von der Position des Umstimmelementes. Gleichzeitig kann die aktuelle Position mit einem Sensor bestimmt werden. Eine *Client Aktor Steuerung (CAS)* regelt die Position von bis zu 4 Umstimmeinrichtungen, dabei nutzt sie eine gespeicherte Kennlinie, die die Position der Umstimmeinrichtung in ein Verhältnis zur Tonhöhe der Pfeife setzt. Diese Kennlinie wird derzeit mithilfe eines, mit der *Zentralen Aktor Steuerung (ZAS)* verbundenen, Stimmgerätes während der Wartung der Orgel aufgezeichnet und für jede Pfeife individuell in der zugehörigen CAS gespeichert. Bei der ZAS handelt es sich um ein Embedded Linux System, welches mithilfe des HMT-Modules, aus

gespielten Tönen, die notwendige Umstimmung der Orgel errechnet. Das HMT-Modul ist ein externes Gerät, welches über eine serielle Verbindung mit der ZAS kommuniziert und die notwendige Umstimmung gegenüber einer gleichstufigen Stimmung errechnet. Weiterhin werden über die ZAS verschiedene Funktionen wie die Wahl der Stimmung oder Wartungsfunktionen, wie zum Beispiel das Aufnehmen der Kennlinien, bereitgestellt.

Während ein Organist nun auf dem Instrument spielt, werden durch die Orgelsteuerung die gespielten Töne mithilfe der MIDI-Schnittstelle[1] an die ZAS übertragen. Mithilfe des HMT-Moduls wird nun bestimmt, welche Pfeifen wie stark umgestimmt werden müssen. Die ZAS übermittelt diese Informationen an die betroffenen CAS, welche die Neupositionierung der Umstimmeinheiten anhand der gespeicherten Kennlinien veranlassen.

In dieser Arbeit sollen Algorithmen zur Bestimmung der Grundfrequenz hinsichtlich ihrer Tauglichkeit zur Bestimmung der Tonhöhe von Orgelpfeifen überprüft werden. Dabei sollen Aspekte der späteren Implementierung der Stimmfunktionalität in der ZAS berücksichtigt werden, so dass zukünftig auf den Einsatz eines externen, zugekauften Stimmgerätes verzichtet werden kann.

[1] MIDI (Musical Instrument Digital Interface) ist ein Standard für die Übertragung von musikalischen Steuersignalen

2 Grundlagen

2.1 Musiktheoretische Grundlagen

Die Intervalle

Mit Intervallen wird in der Musik der Abstand zwischen zwei Tönen bezeichnet. Maßgebend für die Benennung des Intervalls ist hier der Abstand in Halbtonschritten. Die Frequenzverhältnisse der reinen Intervalle lassen sich anhand der Obertonreihe berechnen [2, S. 26 ff.]. Dabei handelt es sich um eine Reihe von Tönen, die jeweils ein ganzzahliges Vielfaches einer Grundfrequenz sind. So besitzt der erste Oberton, auch zweiter Teilton genannt, die doppelte Frequenz des Grundtones. Der Abstand zwischen beiden Tönen wird Oktave genannt. Zwischen dem ersten und dem zweiten Oberton wiederum entsteht ein Intervall mit dem Frequenzverhältnis $\frac{3}{2}$, die Quinte. Analog lassen sich noch viele weitere reine Intervalle herleiten, auf die in dieser Arbeit nicht weiter eingegangen werden soll.

Relative Tonabstände

Um den relativen Abstand zwischen 2 unterschiedlichen Tönen, mit den zugehörigen Grundfrequenzen f_1 und f_2, zu beschreiben, wurde die logarithmische Einheit Cent eingeführt [3]. Entsprechend der Einteilung einer Oktave in 12 Halbtöne wird eine Frequenzverdopplung mit 1200 Cent definiert. Durch folgende Gleichung kann der relative Tonabstand zwischen 2 Tönen berechnet werden:

$$i = 1200 \cdot \log_2 \left(\frac{f_1}{f_2} \right) \ Cent \tag{2.1}$$

Notationsformen

Um einen Ton und dessen Frequenz eindeutig identifizieren zu können, wurden verschiedene Formen der Notation ersonnen. Die wohl bekannteste Form ist die Notenschrift, in der ein Ton durch seine Lage im Notensystem, mit entsprechendem Notenschlüssel und Vorzeichen, eindeutig bestimmbar wird. Um jedoch nur einen Ton zu benennen, existieren viele verschiedene andere Notationsformen. Dabei wird das Tonsystem in Oktavräume eingeteilt, die jeweils mit C beginnen und mit H enden. Jede dieser Oktaven hat einen eigenen Namen wie in Abbildung 2.1 zu sehen ist. Für diese Arbeit sind zwei Notationsformen relevant: die Helmholtznotation und die wissenschaftliche Notation.

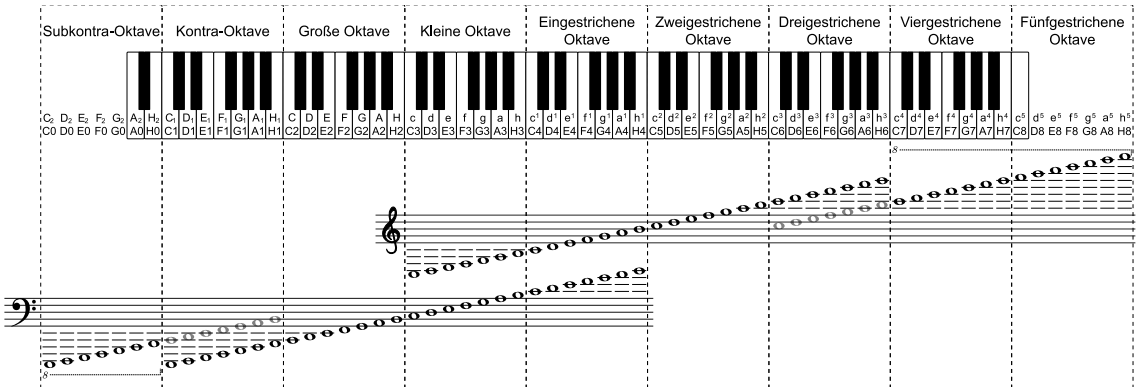


Abbildung 2.1: Tonnamen auf einer Klaviertastatur in Helmholtz- und wissenschaftlicher Schreibweise [4]

Im deutschen Sprachraum ist zur Benennung der Töne die Notation nach Helmholtz verbreitet. Bei dieser Notation werden für alle Oktaven bis einschließlich zur *großen Oktave* Großbuchstaben verwendet, ab der *kleinen Oktave* Kleinbuchstaben. Für die Oktavlagen ab der eingestrichenen Oktave werden nach dem Buchstaben Primen gesetzt. So erhält die eingestrichene Oktave eine Prime, die höheren Oktavlagen entsprechend mehr. Das *Zweigestrichene C* wird so zum Bespiel als c'' notiert. Die Kontra-Oktave wird mit einer, die Subkontra-Oktave mit zwei tiefgestellten Primen gekennzeichnet. Es ist eine verbreitete Schreibweise, anstelle der Primen hoch- beziehungsweise tiefgestellte Ziffern zu verwenden. So wird c'' als c^2 notiert. Um eine klare Abgrenzung zur Fußtonzahl (siehe Abschnitt 2.3) zu gewährleisten, wird in der vorliegenden Arbeit ausschließlich die numerische Helmholtz-Notationsform verwendet.

Im englischen Sprachraum hingegen wird zumeist die wissenschaftliche Notationsweise genutzt. Diese Notation nutzt ausschließlich Großbuchstaben und eine Ziffer, die mit jeder Oktavlage um eins erhöht wird. Die Zählung beginnt hier in der Subkontra-Oktave mit 0. Da sie relativ einfach zu verarbeiten ist, wird sie häufig in der Softwareentwicklung verwendet.

Die Chromatische Tonleiter & Der Kammerton

Das heute verbreitete Tonsystem basiert auf der Einteilung einer Oktave in 12 Halbtonschritte. Bei einer *gleichstufigen Stimmung* sind diese logarithmisch gleichmäßig verteilt und weisen so einen Abstand von je 100 Cent (siehe Abschnitt 2.1) zueinander auf. Die so entstandene Tonleiter nennt man auch *chromatische Tonleiter*.

Mit dem Wissen, dass ein Halbtonschritt der Tonleiter 100 Cent entspricht, lässt sich aus Gleichung 2.1 eine Formel zur Bestimmung der Frequenz beliebiger Tonleitertöne herleiten. Hierbei sei f die gesuchte Frequenz, f_k die Frequenz eines beliebigen Tonleitertons und n der Abstand zwischen dem gesuchten Ton und dem bekannten Ton f_k in Halbtonschritten.

$$i = 1200 \cdot \log_2 \left(\frac{f}{f_k} \right) \; Cent \tag{2.2}$$

$$n = 12 \cdot \log_2 \left(\frac{f}{f_k} \right) \; Halbtonschritte \tag{2.3}$$

Durch Umstellen der Formel nach f erhält man Gleichung 2.4, mit deren Hilfe es möglich ist, für jeden Ton der chromatischen Tonleiter die Frequenz anhand eines Referenztones f_k zu berechnen.

$$f(n) = f_k \cdot 2^{\frac{n}{12}} \tag{2.4}$$

Mithilfe des Kammertons a^1 kann nun die Tonhöhe der Stimmung eines Instruments festgelegt werden. Eine Orgel soll dabei so gestimmt werden, dass die Frequenz des Tones a^1 440Hz entspricht, wenn die Umgebung ihre Jahresmitteltemperatur besitzt [5]. Es ist hier üblich die Formel auf die Tastennummern einer Klaviertastatur zu normieren.

$$f(n) = f_k \cdot 2^{\frac{n-49}{12}} = f_k \cdot (\sqrt[12]{2})^{n-49} \tag{2.5}$$

Die Variable n stellt in dieser Gleichung die Tastennummer des gesuchten Tones dar. Diese orientiert sich an der Tastatur eines Klaviers, dessen tiefster Ton zumeist das Subkontra A (A_2) ist. Dementsprechend besitzt diese Taste die Tastennummer 1. Der Kammerton a^1 ist von diesem Subkontra A 4 Oktaven, zu je 12 Halbtonschritten, entfernt. Dementsprechend lautet seine Tastennummer 49.

2.2 Der Spieltisch einer Orgel

Der Spieltisch einer Orgel besteht im wesentlichen aus einer oder mehreren Klaviaturen (Manual und Pedal), Registerzügen und einem Notenpult. Häufig befinden sich hier außerdem Bedienelemente für Spielhilfen der Orgel wie zum Beispiel Koppeln. Den Tasten bzw. Pedalen der Klaviaturen, beginnend meist mit dem C, sind hierbei jedoch zunächst keinen Pfeifen zugeordnet. Wird eine Taste gedrückt, so erklingt zunächst kein Ton. Erst durch Betätigung der Registerzüge werden die Tasten mit Pfeifen unterschiedlicher Klangfarbe und Tonhöhe zugeordnet(siehe Abschnitt 2.3). Dabei muss der erklingende Ton nicht notwendigerweise der Ton sein, der gedrückt wurde. Beispielsweise erklingt bei einem gezogenen $2'$-Register ein a^1, wenn die Taste A betätigt wird (siehe Abschnitt 2.3).

2.3 Orgelregister

Die einzelnen Orgelpfeifen gleicher Bauart werden in Registern zusammengefasst. Auf diese Weise entstehen Gruppen von Pfeifen gleicher Klangfarbe. Die Register werden weiterhin nach ihrer Tonhöhe unterteilt. Die Bemessung erfolgt dabei anhand der Tonhöhe des tiefsten Tones C und der hierfür notwendigen Pfeifenlänge als offene Labialpfeife (siehe nächster Absatz) in Fuß. Diese Länge nennt man auch *Fußtonzahl*. [6, S. 49, ff.] Ein $8'$-Register weist hierbei die normale Tonhöhe auf, das heißt, ein a^1 besitzt die Grundfrequenz 440Hz. Ein $4'$-Register hingegen ist eine Oktave höher gestimmt, weist also jeweils die doppelte Tonfrequenz auf.

Die Bauarten von Orgelpfeifen können zunächst in 2 wichtige Gruppen unterteilt werden:

- *Labial-* oder *Lippenpfeifen*
- *Lingual-* oder *Zungenpfeifen*

In Labialpfeifen entsteht der Ton durch Schwingungen der Luftsäule im Pfeifenkörper. Bei Lingualpfeifen wird der Ton hingegen durch eine schwingende Metallzunge erzeugt [6, S. 40]. Klanglich sind Lingualpfeifen dabei oft sehr obertonreich.

Im Unterschied zu beispielsweise Klaviersaiten weisen Orgelpfeifen keine nennenswerte Inharmonizität auf. Das bedeutet, dass die Obertöne in einem Pfeifenklang ganzzahlige Vielfache der Grundfrequenz sind und so Rückschlüsse auf diese erlauben [6, S. 53 ff.].

Häufig ist die erklingende Tonhöhe einer Pfeife über der Zeit nicht konstant, sondern schwingt um eine Mittenfrequenz. Dieses Verhalten bezeichnet man als Tremulieren. Die Periodendauer dieser Schwingung kann dabei mehrere Sekunden betragen. Auch Frequenzänderungen von mehr als 50 Cent sind möglich. Die durch den Hörer wahrgenommene Tonhöhe des Tones entspricht jedoch der Mittenfrequenz.

2.4 Pitch und Fundamentalfrequenz

Bei der Grund- oder Fundamentalfrequenz handelt es sich um den Grundton, also die erste Spektrallinie eines Tones. Der Pitch, die wahrgenommene Tonhöhe, kann hiervon abweichen, wenn der Ton inharmonisch ist, also die Frequenz der Obertöne kein ganzzahliges Vielfaches der Grundfrequenz ist.

Da bei Orgelpfeifen keine signifikante Inharmonizität auftritt, entspricht die wahrgenommene Tonhöhe auch der Fundamentalfrequenz, solange der Ton nicht tremuliert.

2.5 Die Fouriertransformation

Bei der Fouriertransformation handelt es sich um eine Signaltransformation, mit der ein Signal aus dem Zeitbereich in den Frequenzbereich transformiert werden kann. Im Frequenzbereich erkennt man unter anderem leicht, aus welchen Frequenzkomponenten bzw. Teiltönen ein Signal besteht. Für ein kontinuierliches Signal ist die Fouriertransformation wie folgt definiert [7, S. 165]:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{2.6}$$

Für ein mit der Frequenz $f_s = \frac{1}{T}$ N-mal abgetastetes Signal wurde weiterhin die diskrete Fouriertransformation (DFT) definiert [7, S. 298].

$$F(n) = \sum_{k=0}^{N-1} f(k)e^{-j\frac{n}{N}2\pi k} \tag{2.7}$$

Zu beachten ist dabei, dass das Ergebnis der DFT wiederum eine diskrete Funktion, das diskrete Spektrum ist. Der Abstand der Spektrallinien beträgt dabei:

$$\Delta f = \frac{1}{N} f_s \tag{2.8}$$

Die DFT kann mittels eines speziellen Algorithmus, dem FFT-(Fast Fourier Transformation)-Algorithmus, in kurzer Zeit berechnet werden.

2.6 Die Autokorrelationsfunktion

Die Autokorrelationsfunktion (AKF) beschreibt die innere Ähnlichkeit eines Signales [7, S. 372 ff.]. Sie wird definiert als:

$$r_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x(t+\tau)dt \tag{2.9}$$

Das globale Maximum der Autokorrelierten findet sich an der Position $r_{xx}(0)$, da in diesem Falle die Ähnlichkeit eines Signales mit sich selbst beschrieben wird. Handelt es sich bei x(t) um ein Signal mit periodischen Komponenten, so weist die Autokorrelierte lokale Maxima an den vielfachen der Periodendauer auf. Ein dem Signal überlagertes Rauschen führt hier zu keinen lokalen Maxima, da es keine Periodizität aufweist. Für diskrete Signale wird die Autokorrelationsfunktion über einem Beobachtungsfenster der

Breite W wie folgt definiert [8]:

$$r_{xx}(\tau) = \sum_{j=t}^{t+W-1-\tau} x_j x_{j+\tau} \tag{2.10}$$

2.7 Die Fensterfunktionen

Bei Fensterung handelt es sich um eine Methode zur Gewichtung der einzelnen Datenpunkte eines diskreten Signales. Die Fensterfunktion weist dabei jedem einzelnen Datenpunkt eine Gewichtung zu. Die Anwendung erfolgt durch eine Multiplikation von Signal und Fensterfunktion im Zeitbereich. Auf diese Weise kann der Leckeffekt[2] stark vermindert werden. Abhängig von der Anwendung, zum Beispiel für den Filterentwurf oder die Spektralanalyse, können unterschiedliche Fensterfunktionen geeignet sein [9, S. 180 ff.]. Mögliche Bewertungskriterien ergeben sich anhand der Spektralfunktion des Fensters und sind zum Beispiel:

- Die Breite des Hauptmaximums
- Die relative Höhe des 1. Nebenmaximums
- Die Sperrdämpfung

2.8 Fehlerbetrachtung der AD-Wandlung

Bevor ein Ton mit den Methoden der digitalen Signalverarbeitung analysiert werden kann, muss zunächst die physikalische Größe Luftdruck in ein elektrisches Signal umgewandelt werden, um dann mit einem AD-Konverter quantisiert zu werden. Die Schallwandlung erfolgt hier mit einem Mikrofon. Das resultierende elektrische Signal wird verstärkt und tiefpassgefiltert, um Frequenzkomponenten oberhalb des hörbaren Bereiches zu entfernen. Hierbei treten verschiedene Effekte auf, die das spätere Messergebnis beeinflussen.

Rauschen und Quantisierung

Durch jedes Bauelement im analogen Signalweg wird dem Signal zusätzliches Rauschen hinzugefügt, bis es schließlich in ein digitales Signal gewandelt wird. Bei der AD-Wandlung wird nun eine Quantisierung vorgenommen, das heißt das analoge Signal wird durch eine Zahl endlicher Genauigkeit dargestellt. Auch diesen Verlust an Information kann man durch Addition eines Rauschsignales beschreiben [10, S. 23 ff.].

[2] Der Leckeffekt beeinflusst das Ergebnis der DFT negativ. Aufgrund des endlichen Beobachtungszeitraums und der periodischen Fortsetzung treten 2 Hauptlinien, sowie weitere Spektrallinien an sämtlichen DFT-Rasterpunkten auf [9, S.276 ff.].

Auf die spätere Analyse wirkt sich dieses Rauschen als Verringerung der Auflösung des Signals aus. Dementsprechend verringert sich auch die spätere Messgenauigkeit.

Abtastratenfehler

Weicht aufgrund von Fertigungstoleranzen oder Alterung der Bauelemente die Abtastrate von ihrem Sollwert ab, so entsteht ein linearer Fehler in Abhängigkeit vom Messergebnis. Um diesen Fehler zu korrigieren, kann eine Messung mit einem Referenzton durchgeführt werden. Aus dem Verhältnis der Referenzfrequenz zum Messergebnis kann die korrekte Abtastrate berechnet werden.

2.9 Stand der Technik

Derzeit gibt es verschiedene Stimmgeräte für Orgelpfeifen am Markt. Dabei sind sowohl kompakte Geräte als auch reine Softwarelösungen für PC, Tabletcomputer oder Mobiltelefone verfügbar. Diese unterscheiden sich hinsichtlich ihres Funktionsumfangs und ihrer Genauigkeit. Unglücklicherweise wird bei vielen Geräten auf die Angabe der Genauigkeit verzichtet. Verglichen wurden daher:

- Korg CA-40
- TLA CTS32-C

Das kompakte Stimmgerät Korg CA-40 wird mit einer Genauigkeit von $\pm$ 1 Cent beworben. Das bisher verwendete Stimmgerät TLA CTS32-C ist nur durch seine Auflösung von 0,1 Cent spezifiziert.

Es gibt weiterhin Stimmgeräte mit automatischer Tonerkennung und solche, bei denen der gespielte Ton manuell am Gerät eingestellt werden kann bzw. muss. Dabei findet sich die Möglichkeit der manuellen Einstellung vor allem an hochwertigen Geräten. Das TLA CTS32-C nutzt die durch den eingestellten Ton vorgegebene Sollfrequenz, um mit einem Bandpassfilter Obertöne und andere Störgeräusche zu unterdrücken.

Weiterhin ist eine große Menge an Algorithmen bekannt, mit denen aus einem diskreten Signal die Grundfrequenz berechnet werden kann. Häufig stammen diese aus dem Bereich der Sprachanalyse, sind für die dortigen Anforderungen optimiert und ermöglichen neben der Erkennung der Grundfrequenz noch weitere Analysen. Die freie Software *Praat*, von Paul Boersma und David Weenink, welche im Bereich der phonetischen Analyse verbreitet ist, enthält unter anderem auch mehrere Algorithmen zur Bestimmung der wahrgenommenen Tonhöhe.

Die meisten Algorithmen zur Bestimmung der Grundfrequenz beziehungsweise des Pitches eines Signals basieren dabei auf der Autokorrelationsfunktion oder der Fourier-

transformation. Beispiele für Algorithmen sind:

- Time-Event-Rate-Detection - Zeitbereich
- McLeod-Algorithmus - Zeitbereich
- YIN-Algorithmus - Zeitbereich
- FFT-Analyse - Frequenzbereich
- Cepstral-Analyse - Frequenzbereich
- YAAPT pitch tracking - Gemischt: Zeit- und Frequenzbereich

3 Algorithmen zur Grundfrequenzbestimmung

3.1 Anforderungen

Das derzeitige Konzept zur Verwendung des Versuchsinstrumentes sieht vor, dass die Orgel während des Spielbetriebes anhand gespeicherter Kennlinien umgestimmt wird. Diese Kennlinien, die die Tonhöhe jeder Pfeife in Abhängigkeit von der Position des Stimmelementes beschreiben, werden während einer Kalibrierung aufgenommen. So stehen für jede Pfeife ca. 11 Sekunden für die Messung der Grundfrequenz zur Verfügung. Begrenzt wird dies durch die maximal für die Kalibrierung eines Registers vorgesehene Zeit von 10 Minuten.

Die erforderliche Genauigkeit richtet sich nach dem menschlichen Gehör. Wenn Töne nacheinander erklingen, ist ab ca. 2 Cent eine Tonänderung wahrnehmbar. Erklingen die Töne jedoch gleichzeitig entstehen Schwebungen. Durch diese kann ein Mensch mit geschultem Gehör auch sehr geringe Tonabweichungen wahrnehmen. Die minimal hörbare Abweichung ist über dem hörbaren Frequenzbereich nicht konstant und vom Hörer abhängig. Aus diesem Grund soll sich die Anforderung für den neuen Algorithmus an bestehenden Stimmgeräten orientieren. Das bisher im Projekt eingesetzte TLA CTS-32-C besitzt eine Auflösung von 0,1 Cent, dies soll die Anforderung an die Genauigkeit für den Algorithmus sein.

Die erforderliche Bandbreite wird zum einen durch den tiefsten Ton eines $32'$-Registers und zum anderen durch das menschliche Gehör bestimmt. Der Ton C, eines $32'$-Registers, schwingt unter Berücksichtigung einer möglichen Verstimmung mit mindestens 14 Hz. Die obere Grenzfrequenz, die Hörgrenze des Menschen, wird mit 20 kHz angegeben.

Da eine Implementierung auf einem Embedded Linux System angedacht ist, sollten die Algorithmen mit typischen Abtastraten verbreiteter Audiohardware zuverlässig arbeiten.

3.2 Überblick und Auswahl der Algorithmen

Die Bestimmung der Grundfrequenz bzw. der wahrgenommenen Tonhöhe ist ein Problem, für das in vielen Jahren diverse Lösungsansätze entwickelt wurden. Entsprechend der Verarbeitung der Daten kann man zwischen 3 Gruppen unterscheiden. Den Algorithmen, welche die Daten im Zeitbereich verarbeiten, denen, die Daten im Frequenzbereich verarbeiten und solchen, die eine Mischform aus beiden nutzen. Ein Großteil

dieser Algorithmen basiert schließlich auf Fouriertransformation oder Autokorrelationsfunktion.

Eine der einfachsten Methoden, aus der Gruppe der *Time-Event Rate Detection* Methoden, zur Bestimmung der Frequenz einer Schwingung, ist den Abstand zwischen den Nullstellen, die sogenannte *Zero-crossing rate*, zu bestimmen. Doch diese Methode kann bereits versagen, sobald Obertöne oder Rauschen zur Grundfrequenz hinzukommen. Leicht zu erkennen ist dies zum Beispiel bei der Funktion $f(x) = 0,9 \cdot sin(x) + 0,1 \cdot sin(20x)$ [11, S. 4 f.] die im Bereich von π mehrere Nullstellen dicht nebeneinander aufweist.

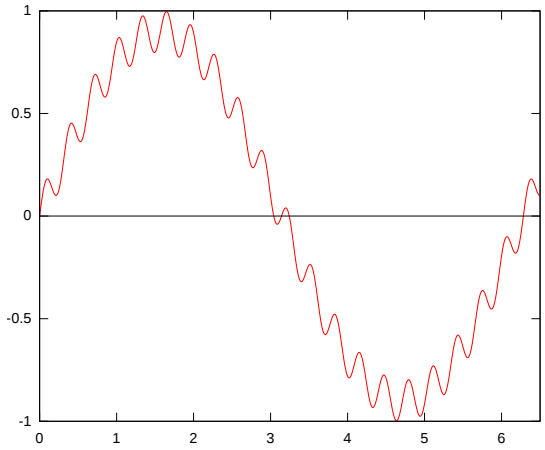


Abbildung 3.1: Funktion $f(x)$ mit mehreren Nulldurchgängen in der Nähe von π

Im Rahmen dieser Arbeit wurden 3 Methoden zur Bestimmung der Grundfrequenz näher untersucht. Kriterien waren hierbei die Aktualität der Veröffentlichung und die Verbreitung beziehungsweise die Existenz frei verfügbarer Implementierungen. Zur näheren Betrachtung ausgewählt wurden:

- Der Autokorrelationsalgorithmus *ac* der Software *Praat*
- Die Frequenzbestimmung auf Basis zweier zeitverschobener FFT
- Die McLeod-Methode [8]

Die Algorithmen wurden hinsichtlich ihrer Genauigkeit bei sinusförmigen Signalen unterschiedlicher Abtastraten untersucht. Weiterhin wurde untersucht, wie Umgebungsgeräusche die Messergebnisse beeinflussen.

3.3 Der Autokorrelationsalgorithmus

Die Bestimmung der Grundfrequenz mithilfe der Autokorrelationsfunktion erfolgt analog dem in der Sprachanalysesoftware *Praat* “ac“-Algorithmus zur Pitchdetection [12].

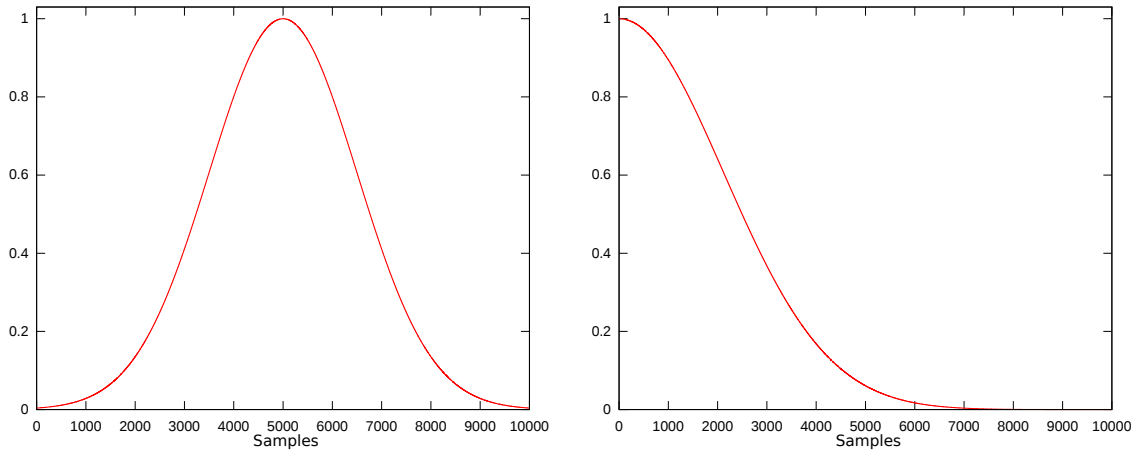


Abbildung 3.2: Gauß-Fensterfunktion und ihre normierte Autokorrelationsfunktion

Zunächst wird das Signal durch eine Fensterfunktion gewichtet. Anschließend werden die normierten Autokorrelationsfunktionen von gefenstertem Signal und Fensterfunktion berechnet. Nun wird an jeder Stelle der Quotient aus beiden gebildet. Als Fensterfunktion wurde entsprechend [12] das Gauß-Fenster gewählt. Eine Erprobung mit einem Kaiserfenster [9, S. 185 ff.] brachte keine Verbesserung der Messergebnisse und floss daher in die spätere Untersuchung nicht ein. Als Koeffizient α des Gaußfensters wurde 0,5 gewählt.

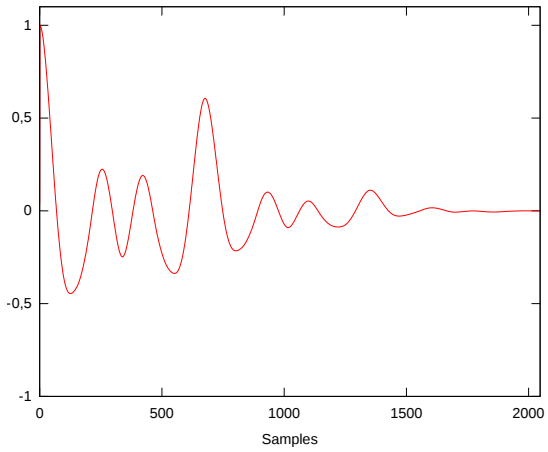


Abbildung 3.3: Normierte Autokorrelationsfunktion des Tones C im Register *Viol di Gamba* $8'$

Im so entstandenen Signal erfolgt nun die Suche nach Maximalwerten. Die Auswertung, die in [12] vorgestellt wird, wird zum Zwecke der Erkennung von Pfeifentönen stark vereinfacht, da eine Unterscheidung zwischen stimmhaften und nicht stimmhaften Konsonanten nicht erforderlich ist. Auch auf die Messung des *harmonics to noise ratio* kann verzichtet werden. So wird anstelle der in beschriebenen Prozedur ein einfacher Schwellwert für die minimale Höhe eines lokalen Maximums angesetzt, das zur Auswertung herangezogen wird. Aus dem zwischen Abstand zwischen 2 so gefundenen lokalen Maximalwerten ergibt sich die Frequenz.

$$f_g = \frac{\Delta\tau}{f_{sample}} \tag{3.1}$$

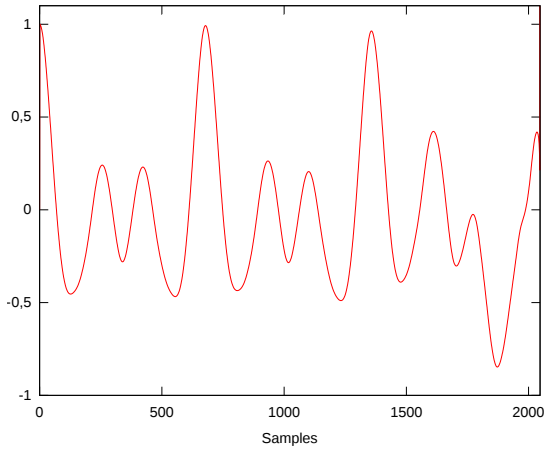


Abbildung 3.4: Ergebnis der Division der Autokorrelationsfunktionen des Tones C im Register *Viol di Gamba* $8'$ und der Fensterfunktion

3.4 Der Fourieralgorithmus

Mittels der Fouriertransformation kann das Frequenzspektrum des Eingangssignals berechnet werden. Die Partialtöne des Signales werden dabei als Spitzen dargestellt. Die Genauigkeit ist hierbei jedoch auf die Auflösung der Fouriertransformierten beschränkt. Diese beträgt $\Delta f = \frac{f_{sample}}{N}$, wobei f_{sample} die Abtastrate des Signals und N die Anzahl der transformierten Samples ist. Bei einer Abtastfrequenz von 44,1 kHz und 2048 ausgewerteten Samples ergibt sich die Auflösung somit zu 21,5Hz. Da der absolute Frequenzabstand zwischen zwei Halbtönen in Hertz mit zunehmender Tonhöhe größer wird, steigt die relative Genauigkeit in Cent an. Mithilfe von Gleichungen 2.1 und 2.8 lässt sich bestimmen, ab welcher Frequenz die erforderliche Genauigkeit erreicht wird:

$$\begin{aligned} \Delta f_{rel}(f) &= 1200 \cdot \log_2 \left(\frac{f - \frac{\Delta f}{2}}{f} \right) \; Cent \\ &= 1200 \cdot \log_2 \left(1 - \frac{f_{sample}}{2Nf} \right) \; Cent \end{aligned} \tag{3.2}$$

Zu beachten ist, dass das Ergebnis hier immer eine negative Abweichung ist, da diese, aufgrund der logarithmischen Funktion, betragsmäßig größer ist als die Abweichung in positiver Richtung. Das Ziel einer Auflösung von weniger als 0,1 Cent Abweichung wird über dem gesamten Frequenzbereich verfehlt. Erst mit einer extrem hohen Fenstergröße von über 27 Millionen Samples könnte die Anforderung erfüllt werden.

Um dieses Problem zu lösen, wird eine allgemein bekannte Analysemethode angewen-

det. Es werden 2 Fouriertransformierte erstellt. Dabei wird die erste Fouriertransformierte wird aus Folge der Samples $a_0 = (x_0, ..., x_{N-1})$ berechnet. Die zweite verschobene Fouriertransformierte nutzt die Folge der Samples $a_1 = (x_n, ..., x_{n+N-1})$. Beide Folgen a_0 und a_1 müssen zur Vermeidung von Leck-Effekten mit einer Fensterfunktion gewichtet werden. Das Kaiserfenster [9, S. 185 ff.] erwies sich bei einer Erprobung des Algorithmus als gut geeignet. Als Koeffizient β (benannt entsprechend [9]) wurde 8 gewählt.

Im Anschluss wird das Maximum des Betrages der Fouriertransformierten ermittelt und das Argument der komplexen Zahl an dieser Position bei beiden Fouriertransformierten gebildet. Die Differenz der beiden Winkel entspricht der Winkeländerung der gesuchten Frequenz während der Verschiebungszeit von k Samples. Folglich kann die gesuchte Frequenz leicht bestimmt werden:

$$f_g = f_{sample} \cdot \frac{\Delta\varphi}{k \cdot 2\pi} \tag{3.3}$$

Da auf diese Art und Weise immer der stärkste Teilton detektiert wird, handelt es sich bei der ermittelten Frequenz möglicherweise um die Frequenz eines Obertones. Um den Grundton zu ermitteln, wird die Tatsache genutzt, dass die Tonlage des analysierten Signales bekannt ist. Wird ein dominanter Oberton, wie zum Beispiel in Abbildung 3.5, detektiert, so ist die ermittelte Frequenz ein ganzzahliges Vielfaches der Grundfrequenz. Durch Division mit der Sollfrequenz des Tones ist bekannt, um welchen Teilton des Klanges es sich handelt. Da der n-te Teilton bei Orgelpfeifen immer mit der n-fachen Grundfrequenz schwingt, kann durch einfache Division der Grundton mit hoher Genauigkeit ermittelt werden. So wird zuletzt die ermittelte Frequenz durch die Nummer des detektierten Teiltones dividiert.

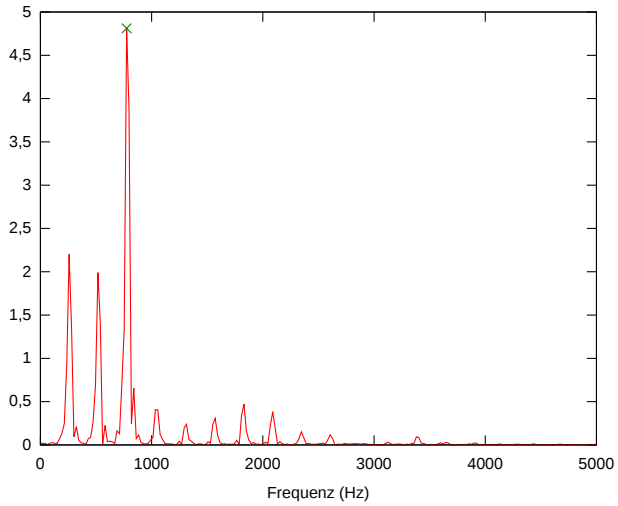


Abbildung 3.5: Spektrum des Tones c' aus dem Register *Viol di Gamba* $8'$

3.5 Der McLeod Algorithmus

Zuletzt wurde der McLeod-Algorithmus zur Untersuchung ausgewählt. Bei diesem Algorithmus handelt es sich um einen Algorithmus zur *pitch detection*. Er wurde im Jahr 2005 detailliert von McLeod und Wyvill beschrieben [8]. Dieser Algorithmus basiert auf der quadratischen Differenzfunktion:

$$d_t(\tau) = \sum_{j=0}^{N-1} (x_j - x_{j+\tau})^2 \tag{3.4}$$

Durch Ausmultiplizieren von $(x_j - x_{j+\tau})^2$ wird erkennbar, dass die quadratische Differenzfunktion auch die Autokorrelationsfunktion enthält: $x_j^2 + x_{j+\tau}^2 \underline{-2x_j x_{j+\tau}}$.

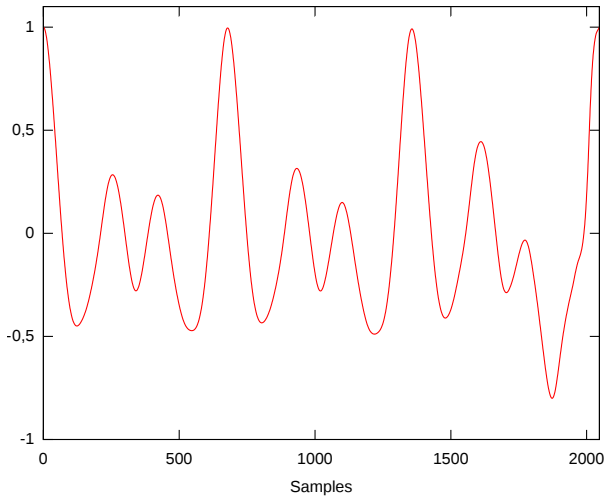


Abbildung 3.6: Normierte quadratische Differenzfunktion des Tones C aus dem Register *Viol di Gamba* $8'$

Wie auch im Autokorrelationsalgorithmus erfolgt eine Normierung der Funktion und die anschließende Suche nach relevanten Maxima. Zur Nutzung dieses Algorithmus müssen 2 Parameter festgelegt werden. Zum einen muss die Breite des Beobachtungsfensters so gewählt werden das zumindest 2 Perioden eines Signales mit der minimal zu messenden Signalfrequenz erfasst werden können, zum anderen ist der konstante Wert k zur Bestimmung des Schwellwertes anzugeben. Typische Werte liegen hier im Bereich $k = 0,8...1,0$.

Der Ablauf des Algorithmus:

1. Berechne die diskrete normierte quadratische Differenzfunktion (NQDF)
2. Finde alle lokalen Maximalwerte der diskreten NQDF
3. Interpoliere alle Maxima der kontinuierlichen NQDF
4. Verwirf die Maxima, welche kleiner als der Schwellwert sind

5. Berechne den mittleren, zeitlichen Abstand zwischen den Maxima und bilde den reziproken Wert

Dabei ist zu beachten, dass aufgrund des endlich langen Beobachtungsfensters die letzten Werte der NQDF stark verfälscht sind (siehe Abbildung 3.6). Aus diesem Grund wurden in der Implementierung nur Maxima bis zu einer Position von 90% der Fensterlänge berücksichtigt.

3.6 Interpolation und Maximalwertsuche

Sowohl der Autokorrelation- als auch der McLeod-Algorithmus erfordern die Auswertung einer diskreten Funktion. In beiden Fällen müssen die lokalen Maxima der Funktion bestimmt werden. Dabei ist der Messfehler direkt von der Präzision des gefundenen Maximums abhängig. Zunächst kann das Maximum nur auf den ganzzahligen Vielfachen der Abtastperiode bestimmt werden. Die Genauigkeit des Messergebnisses ist hierdurch stark eingeschränkt, bei einer Frequenz von 440Hz und einer Abtastrate von 48kHz ergäbe sich beispielsweise eine Abweichung von 3,64Hz oder ca. 14 Cent. Um die Genauigkeit zu erhöhen, ist eine Interpolation zwischen den bekannten Werten erforderlich. Zu diesem Zweck wurden 2 Verfahren getestet, die Interpolation mit einer quadratischen Gleichung und die Interpolation mithilfe der *Whittaker-Shannon*-Funktion.

3.6.1 Quadratische Interpolation

Die erste implementierte Abtastmethode ist die quadratische Interpolation. Aus dem lokalen Maximum sowie den 2 nebenliegenden Datenpunkten werden die Koeffizienten der Gleichung

$$y(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 \tag{3.5}$$

bestimmt. Mithilfe der Ableitung der Gleichung 3.5

$$y'(x) = a_1 + 2 \cdot a_2 \cdot x = 0 \tag{3.6}$$

wird die Position des Maximums bestimmt.

$$x = -\frac{a_1}{2a_2} \tag{3.7}$$

3.6.2 Brent-Algorithmus und Whittaker-Shannon-Interpolation

Aus [13, S. 402 ff.] ist ein Algorithmus zur Suche lokaler Maxima in einer kontinuierlichen Funktion bekannt, der Brent-Algorithmus. Hierbei erfolgt die Annäherung einer

kontinuierlichen Funktion mithilfe der Whittaker-Shannon-Interpolation. Auf Grundlage der Gleichung [12, S.107]

$$x(t) = \sum_{n=-\infty}^{\infty} x_n \cdot sinc\left(\frac{t - t_n}{\Delta t}\right) \tag{3.8}$$

wird eine Interpolation des Funktionswertes zwischen den Abtastzeitpunkten vorgenommen. Hierfür wird der normierte Kardinalsinus verwendet:

$$sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{3.9}$$

Die Implementierung verwendete die 50 umliegende Datenwerte (x_n). Diese wurden entsprechend [12] mit einem Gaußfenster gewichtet.

Die Interpolationsfunktion wird nun als Argument für den Brentalgorithmus genutzt. Das gefundene lokale Maximum, sowie dessen umliegende Punkte wurden der Brentfunktion übergeben. Als Rückgabewert lieferte die Funktion den Wert und die Position des interpolierten Maximums.

Da die Implementierung des Algorithmus aus [13] nicht ohne Lizenzierung für kommerzielle Zwecke verwendet werden darf, kann alternativ die Implementierung aus der *GNU Scientific Library* verwendet werden. Für diese Untersuchung hingegen wurde der Originalcode aus [13] verwendet.

3.7 Implementierung

Die vorgestellten Algorithmen wurden zu Testzwecken in der Programmiersprache C implementiert. Eingesetzt wurde der Compiler gcc in der Version 4.7.2 für die Zielplattform *x86_64-linux-gnu*. Alle Rechenoperationen wurden mit dem Datentyp double (Genauigkeit ca. $1,1 \cdot 10^{-16}$) durchgeführt.

Die Implementierung erfolgte dabei - mit Ausnahme des FFT-Algorithmus - aus Zeitgründen nicht optimiert. So ließe sich eine diskrete Faltung mithilfe einer FFT schneller realisieren. Aus diesem Grund ließen die Untersuchungen keine Auswertung der Speicheranforderungen oder der notwendigen Verarbeitungszeit zu. Für Fouriertransformationen wurde die Bibliothek FFTW[3] in der Version 3.3.2 verwendet. Um den Zugriff auf die Audiodateien zu realisieren, wurde weiterhin die C Bibliothek Libsndfile genutzt. Grafische Oberflächen wurden mit der Grafikbibliothek Qt realisiert.

[3] http://www.fftw.org

Um die Untersuchung durchzuführen, wurden zusätzlich zu den beschriebenen Algorithmen zusätzliche Programme und Scripts entwickelt:

- *analyser* - Auswertung von wave-Dateien
- *hz2cent* - Auswertung der von *analyser* erzeugten Dateien
- *rt_analyser* - Auswertung von mikrofonierten Signalen
- *wave_gen* - Grafischer Generator für obertonhaltige wave-Dateien
- *wave_gen_cmd* - Kommandozeilenbasierter Generator für wave-Dateien
- *f_analyser* - DFT-Analyseprogramm für wave-Dateien
- *scan_script* - Automatisierte Untersuchung aller wave-Dateien im Verzeichnis
- *gen_script* - Automatisierte Erstellung und anschließende Analyse von wave-Dateien

Die Quelltexte aller verwendeten Algorithmen und Programme befinden sich auf der CD im Anhang.

Analyser

Das Programm *analyser* dient zur Analyse der Grundfrequenz in wave-Dateien gespeicherter Signale. Dabei werden immer die ersten N Samples der Datei ausgewertet. Über Kommandozeilenparameter ist das Programm steuerbar:

- -s Spezifizierung der gewünschten Fensterbreite
- -o Zieldatei für die Messergebnisse (Default: Messung.txt)
- -i Dateiname der zu analysierenden Datei

Nach dem Aufruf öffnet das Programm eine wave-Datei und liest die mit dem Parameter -s übergebene Zahl an Samples ein. Diese werden nun mit allen verfügbaren Algorithmen analysiert. Die Messergebnisse werden, gemeinsam mit den Dateinamen der wave-Datei, in die mit dem Parameter -o übergebene Textdatei angehängt. Sollte die Datei nicht existieren, so wird sie angelegt. Weiterhin werden folgende Dateien erstellt:

- *korrelation_praat_ac_** - Enthält die Autokorrelationsfunktion des Autokorrelationsalgorithmus, * enthält Informationen über die eingesetzte Fensterfunktion und den Interpolationsalgorithmus
- *erg_praat_ac_** - Enthält die normierte Autokorrelationsfunktion des Autokorrelationsalgorithmus,* enthält Informationen über die eingesetzte Fensterfunktion und den Interpolationsalgorithmus
- *fft_dif* - Enthält das ermittelte Spektrum des FFT-Algorithmus
- *fft_dif_max* - Enthält die Position des Ermittelten Maximums des FFT-Algorithmus
- *mcleod_** - Enthält die normierte quadratische Differenzfunktion der Datenwerte, * informiert über die verwendete Interpolationsmethode

Hz2cent

Das Programm *Hz2cent* öffnet die von *analyser* erstellte Datei Messung.txt. Die gespeicherten Frequenzen und Messwerte werden eingelesen. Anschließend wird die absolute Abweichung $f - f_{soll}$, die relative Abweichung $\frac{f}{f_{soll}}$ und die Abweichung in Cent jeweils in eine eigene Datei geschrieben. Damit die Auswertung korrekt funktioniert, muss der Dateiname der ursprünglich analysierten Datei die korrekte Frequenz analog der Form “100Hz.wav“ enthalten.

Rt_analyser

Das Programm *rt_analyser* greift unter Nutzung der ALSA-Bibliothek auf die vorhandene Audiohardware zu und analysiert fortlaufend das Mikrofoneingangssignal. Dabei kann durch Betätigung der Zifferntasten der verwendete Algorithmus ausgewählt werden:

- Mode: 1 - McLeod - Parabelinterpolation
- Mode: 2 - McLeod - Brent-Algorithmus
- Mode: 3 - Autokorrelation - Parabelinterpolation
- Mode: 4 - Autokorrelation - Brent-Algorithmus
- Mode: 5 - FFT-Algorithmus

Während der Nutzung des Programmes können die Frequenzen der Töne E, A, D, G und H über die entsprechenden Buchstabentasten auf der Tastatur ausgewählt werden. Mittels der Tasten + und - kann die Oktavlage des Tones verschoben, also die Frequenz verdoppelt bzw. halbiert werden. Liegt der Ton in einem Bereich von $\pm 50\ Cent$ um die eingestellte Frequenz, erfolgt zusätzlich zur Anzeige von Frequenz und gewähltem Algorithmus die Einblendung einer Skala, die die Entfernung zum Zielton visualisiert.

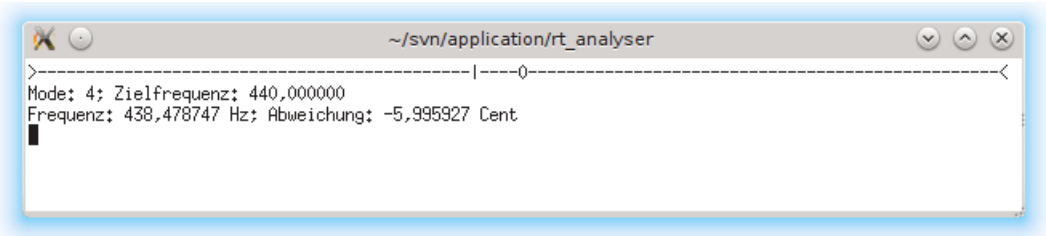


Abbildung 3.7: Das Programm *rt_analyser* zur Messung von mikrofonierten Signalen

Wave_gen

Das Programm *wave_gen* dient zum Erzeugen von wave-Dateien. Es lassen sich neben der Abtastrate und der Dateilänge auch die Lautstärke, die Grundtonfrequenz sowie 6 frei wählbare Obertöne einstellen. Dabei ist das Frequenzverhältnis nicht auf ganzzahlige Vielfache der Grundfrequenz beschränkt. Ebenso ist eine Phasenverschiebung für jeden Teilton separat einstellbar. Zusätzlich kann dem Signal weißes Rauschen hinzugefügt werden. Während der Nutzung wird eine Periode der Grundfrequenz des resultierenden Signales permanent dargestellt und kontinuierlich aktualisiert.

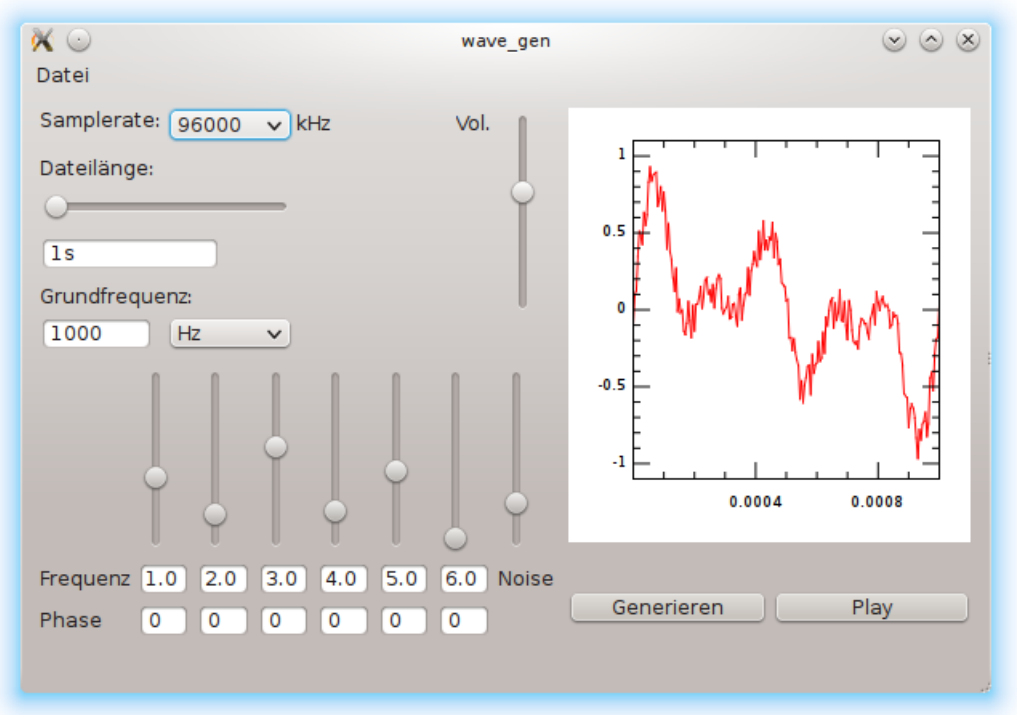


Abbildung 3.8: Das Programm *wav_gen* zur Erzeugung von Testsignalen

Wave_gen_cmd

Das Programm *Wave_gen_cmd* ist in der Lage, kommandozeilengesteuert Wave-Dateien zu erzeugen, die ein Sinussignal enthalten. Es unterstützt dabei die folgenden Argumente:

- -o Name der Ausgabedatei
- -f Spezifizierung der Frequenz
- -s Spezifizierung der gewünschten Samplerate

F_analyser

Das Kommandozeilenprogramm *F_analyser* öffnet die ihm übergebene wave-Datei und liest eine im Quelltext definierte Anzahl von Samples ein. Dieser werden im Anschluss mithilfe der diskreten Fouriertransformation analysiert. Die Ergebnisse werden in der Datei *output.txt* gespeichert.

Scan_script

Bei der Datei *scan_script* handelt es sich um ein Bash-Script, mit dessen Hilfe alle wave-Dateien in einem Verzeichnis automatisiert analysiert werden. Dabei werden verschiedene Fensterbreiten genutzt. Folgende Aufgaben werden dabei umgesetzt:

- Erstellung eines Verzeichnisses für jede vorgegebene Fensterbreite
- Erstellung eines Verzeichnisses für jeden einzelnen Auswertevorgang
- Analyse aller wave-Dateien im Verzeichnis mit allen vorgegeben Fensterbreiten
- Bestimmung der Abweichung mit Hilfe des Programms *Hz2cent*
- Sortieren der Ergebnisse

Gen_script

Die Bash-Scriptdatei *gen_script* erlaubt die automatisierte Erstellung der für den Test benötigten Audiodateien durch Einsatz des Programms *Wave_gen_cmd*. Dabei werden für alle vorgegebenen Abtastraten berücksichtigt (siehe Abschnitt 4.1). Die Dateien gleicher Abtastrate werden in einem Ordner gespeichert. Nach der Erzeugung der Audiodateien wird die Datei *scan_script* aufgerufen.

4 Vergleich

4.1 Testumgebung

Ziel der Untersuchung der in Abschnitt 3 vorgestellten Algorithmen war eine Betrachtung hinsichtlich der Genauigkeit, der Störsicherheit bei Nebengeräuschen und der Funktionalität mit realen Pfeifentönen einer Orgel.

Zunächst erfolgte eine Vermessung von computergenerierten Dateien mit unterschiedlichen Abtastraten. Diese enthielten jeweils nur eine Sinusschwingung bekannter Frequenz, wobei die Amplitude 80% der Vollaussteuerung betrug. Auf Grundlage dieser Dateien sollte eine Aussage zur Genauigkeit der Algorithmen bei unterschiedlichen Fensterlängen, Abtastraten und Signalfrequenzen getroffen werden. Alle Messungen wurden auf einem Computer mit *Intel Core i7* Prozessor unter Debian Linux in der Version 7.3 durchgeführt.

Zur Bestimmung der Frequenzabhängigkeit wurden bei folgenden Referenzfrequenzen gemessen: 14Hz, 30Hz, 40Hz, 50Hz, 60Hz, 70Hz, 80Hz, 90Hz, 100Hz, 200Hz, 300Hz, 400Hz, 500Hz, 600Hz, 700Hz, 800Hz, 900Hz, 1kHz, 2kHz, 3kHz, 4kHz, 5kHz, 6kHz, 7kHz, 8kHz, 9kHz, 10kHz, 11kHz, 12kHz, 13kHz, 14kHz, 15kHz, 16kHz, 17kHz, 18kHz, 19kHz, 20kHz.

Des Weiteren wurden folgende Frequenzen berücksichtigt:

- 440Hz - Kammerton a^1
- 32,7032Hz - Tiefster Ton (C) eines $16'$-Registers
- 4186,01Hz - Ton c^5
- 261,626Hz - Mittleres C (c^1)

Es wurden dabei folgende Abtastraten verwendet:

- 44,1kHz
- 48kHz
- 96kHz
- 192kHz

Alle diese Abtastraten werden von handelsüblichen Soundkarten unterstützt. Niedrigere Standardabtastraten, wie 8kHz oder 22,05kHz, verletzen das Abtasttheorem für die geforderte Bandbreite. Alle Dateien wurden jeweils mit folgenden Fenstergrößen untersucht:

- 512 Samples
- 1024 Samples
- 2048 Samples
- 4096 Samples
- 8192 Samples
- 16384 Samples

Mit Blick auf eine später optimierte Implementierung wurden für die Fenstergröße ausschließlich Potenzen von 2 gewählt.

Um zu überprüfen, inwieweit Obertöne die Analyse negativ beeinflussen, wurden bei ausgewählten, über die gesamte Bandbreite verteilten, Frequenzen weiterhin die Signalformen Sägezahn und Rechteck untersucht.

Zuletzt erfolgte die Untersuchung von an einer realen Orgel aufgenommenen Pfeifentönen. Die Aufnahme erfolgte unter Einsatz der folgenden Hard- und Software:

- Messmikrofon: Behringer ECM8000
- Vorverstärker: Mischpult Behringer Eurorack 802A
- Soundkarte: RME Multiface
- Audiosoftware: Audacity 2.0.1
- Aufzeichnungsformat: wave-Datei (PCM)
- Auflösung: 16Bit

Es wurden jeweils die Töne A und C aller Oktaven sowie der höchste Ton eines Registers aufgenommen. Untersucht wurden dabei folgende Register:

- Subbass-$16'$
- Prinzipal-$8'$
- Oboe-$8'$
- Salicional-$8'$
- Gedackt-$8'$
- Spitzoktave-$4'$
- Waldflöte-$2'$

Dieses Spektrum an Registern ist im wesentlichen mit der Disposition[4] der Versuchsorgel vergleichbar und enthält zusätzlich auch ein Zungenregister (Oboe-$8'$). Nicht zur Verfügung stand ein Register *Terz* $1^3/_5'$.

[4] Die Disposition (von lat. disponere = anordnen) beschreibt den Aufbeu einer Orgel mit Blick auf die Namen und Tonhöhen der Register. Weiterhin werden Werkeinteilung, Spielhilfen, Trakturart und Windladensystem beschrieben. [6, S. 174]

Weiterhin wurden aus dem Register *Cornet 2-fach* die drei Pfeifen höchster Tonhöhe untersucht. Auf diese Weise sollte getestet werden, ob auch die Erkennung von hochfrequenten Tönen zuverlässig funktioniert.

Während der Aufnahme wurde gleichzeitig die Tonhöhe mithilfe eines Stimmgerätes des Typs TLA CTS32-L bestimmt und dokumentiert. Eine Bewertung der Genauigkeit der Algorithmen kann jedoch bei diesen Aufnahmen aufgrund der unbekannten Messgenauigkeit des Stimmgerätes und des Tremulierens vieler Pfeifen nicht vorgenommen werden. Stattdessen soll bewertet werden, ob der Ton trotz Störgeräuschen korrekt erkannt werden konnte. Dabei sind auf den Aufnahmen neben dem Pfeifenton das Betriebsgeräusch des Orgelmotors und bisweilen weitere starke Störgeräusche enthalten.

4.2 Ergebisse: Autokorrelationsalgorithmus

Wie die Auswertung der Messergebnisse zeigte, ist die Genauigkeit über große Teile des Frequenzbereiches in keiner der getesteten Konfigurationen ausreichend, um den Anforderungen zu entsprechen.

Das Messergebnis ist abhängig von Abtastrate, der Breite des auszuwertenden Datenfensters, der gewählten Methode zur Ermittlung lokaler Maxima und der Frequenz, welche zu bestimmen ist. Es wurde festgestellt, dass die obere Grenzfrequenz, bis zu der der Algorithmus Töne zuverlässig erkennt, durch die Abtastrate bestimmt wird (Abbildung 4.1). Steigt diese, so erweitert sich der Messbereich nach oben hin, wobei jedoch, bei konstanter Anzahl berücksichtigter Datenwerte, weniger Signalperioden beobachtet werden und die untere Grenzfrequenz ebenfalls nach oben verschoben wird.

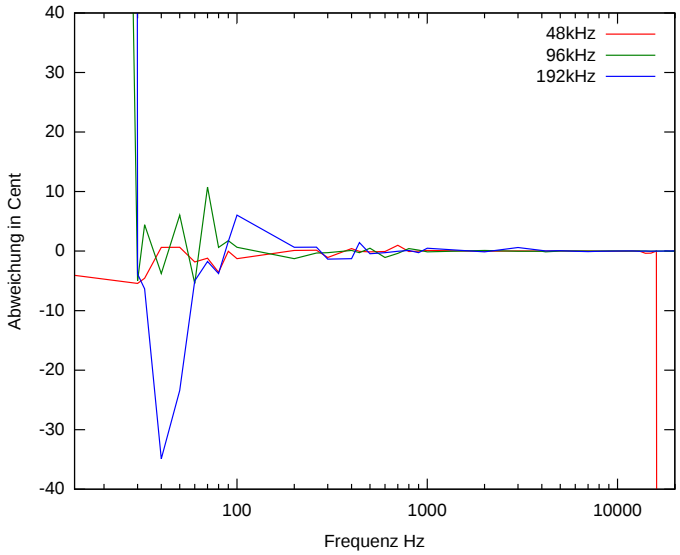


Abbildung 4.1: Messgenauigkeit des AKF-Algorithmus in Abhängigkeit von der Abtastrate bei Parabelinterpolation und einer Fensterbreite von 16384 Samples

Frequenz	Messergebnis FFT-Algorithmus
Sägezahn 14Hz	13,956065Hz
Sägezahn 100Hz	99,909896Hz
Sägezahn 1kHz	1000,073858Hz
Sägezahn 10kHz	9999,956616Hz
Rechteck 14Hz	14,000015Hz
Rechteck 100Hz	147,524068Hz
Rechteck 1kHz	1000,003574Hz
Rechteck 10kHz	9999,956618Hz

Tabelle 4.1: Messergebnisse für obertonhaltige Signalformen unter Nutzung der AKF-Methode bei einer Abtastrate von 48kHz und einer Fensterbreite von 16384 Samples und Parabelinterpolation

Zwischen den Ergebnissen mit unterschiedlichen Methoden zur Lokalisierung der Maxima bestehen nur geringe Unterschiede (Abbildung 4.2). So ist bei der Berechnung nach der Brentmethode die obere Grenzfrequenz etwas geringer. Die Abweichungen zwischen den Messwerten, bei verschiedenen Abtastraten und Fenstergrößen sind im Vergleich zur gesamten Abweichung vom realen Wert nicht signifikant.

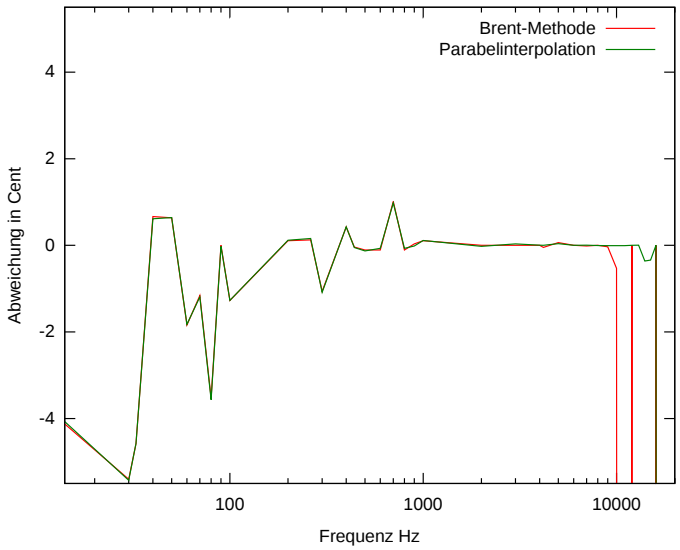


Abbildung 4.2: Messgenauigkeit des AKF-Algorithmus in Abhängigkeit von der Interpolationsmethode bei einer Abtastrate von 48kHz und einer Fensterbreite von 16384 Samples

Eine starke Abhängigkeit besteht zwischen der gewählten Fensterbreite und der Genauigkeit der Messergebnisse. Wie man in Abbildung 4.3 sieht, steigt die Ergebnisgenauigkeit beim Erhöhen der Fensterbreite gravierend an. Des Weiteren empfiehlt [12] für eine reine Grundfrequenzmessung die Fensterbreite wenigstens so groß zu wählen, dass 3 Signalperioden der minimal zu messenden Frequenz im Beobachtungsfenster dargestellt werden können.

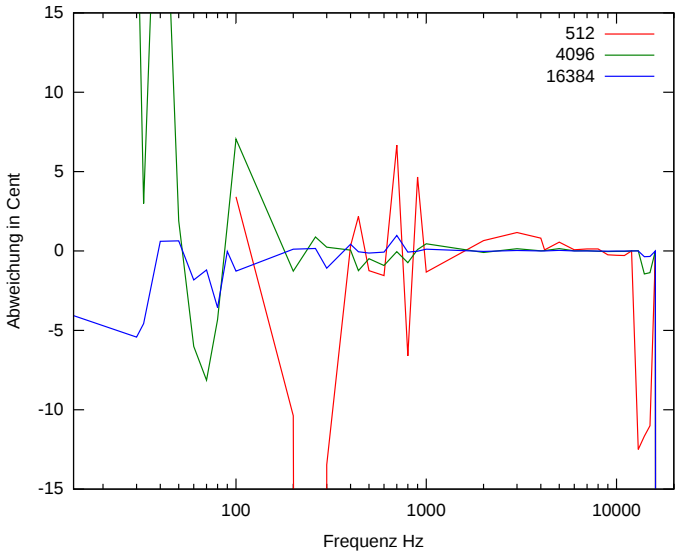


Abbildung 4.3: Messgenauigkeit des AKF-Algorithmus in Abhängigkeit von der Fensterbreite bei einer Abtastrate von 48kHz und einer Fensterbreite von 16384 Samples

Die Erkennungsrate bei der Untersuchung der stark gestörten Aufnahmen von Orgeltönen war gut. Nur in seltenen Fällen, wenn laute Störgeräusche oder Frequenzkomponenten die nicht ganzzahlige Vielfache der Signalfrequenz waren aufgezeichnet wurden, wurde ein Ton nicht korrekt erkannt.

4.3 Ergebisse: FFT-Algorithmus

Der FFT-Algorithmus arbeitet mit sehr hoher Präzision, jedoch muss die Signalfrequenz oberhalb einer unteren Grenzfrequenz liegen. Diese untere Grenzfrequenz ist dabei von Fenstergröße und Abtastrate abhängig. Je größer das Beobachtungsfenster wird, desto weiter verschiebt sich die Grenzfrequenz gegen 0Hz. Abbildung 4.5 zeigt dies exemplarisch für eine Abtastrate von 48kHz.

Da die Auflösung der Fouriertransformation sinkt, wenn die Abtastrate steigt ohne dass die Anzahl der Datenwerte erhöht wird, sinkt auch die Genauigkeit bei steigender Abtastrate. Die untere Grenze, bis zu er die Abtastrate verringert werden kann, wird jedoch durch die Nyquist-Grenze definiert. Folglich muss für eine maximale Signalfrequenz von 20kHz die minimale Abtastfrequenz größer als 40kHz sein.

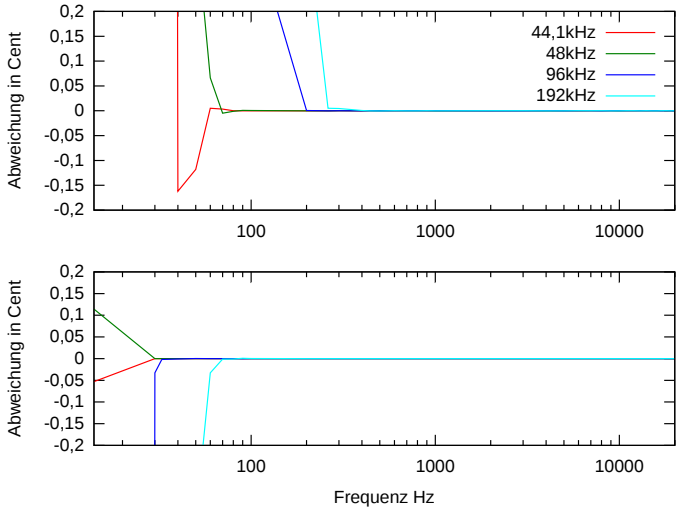


Abbildung 4.4: Messgenauigkeit des FFT-Algorithmus in Abhängigkeit von Frequenz und Abtastrate mit 4096 (oben) und 16384 (unten) Datenwerten

Wie man in Abbildung 4.4 erkennt, ist ab einer unteren Grenzfrequenz die Messungenauigkeit beinahe null. Aufgrund der Fensterung der Messwerte kommt es jedoch bei sehr niedrigen Werten zu Messungenauigkeiten.

Die Messung mit obertonhaltigen Signalformen zeigte bei den verschiedenen Fensterbreiten keine signifikanten Abweichungen zu den bei der Analyse von Sinustönen gleicher Frequenz aufgenommenen Messwerten. Jedoch zeigte die Untersuchung der Aufnahmen von Orgelpfeifen, dass in vielen Fällen Obertöne erkannt wurden, zum Beispiel bei Zungenregistern, bei denen Obertöne dominant waren. Die Frequenzbestimmung dieser Töne erfolgte jedoch mit hoher Genauigkeit. In sehr wenigen Fällen wurden auch Störgeräusche als Ton erkannt. Die Tonerkennung könnte mit einem Bandpassfilter verbessert werden.

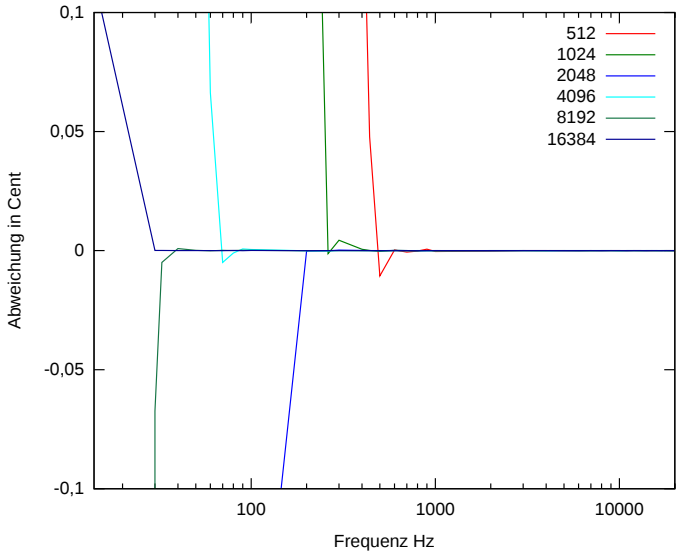


Abbildung 4.5: Messgenauigkeit des FFT-Algorithmus in Abhängigkeit von der Anzahl verwendeter Datenwerte bei einer Abtastrate von 48kHz

Frequenz	Messergebnis FFT-Algorithmus
Sägezahn 14Hz	14,000745Hz
Sägezahn 100Hz	100,000003Hz
Sägezahn 1kHz	1000,0Hz
Sägezahn 10kHz	10000,0Hz
Rechteck 14Hz	14,001449Hz
Rechteck 100Hz	100,000003Hz
Rechteck 1kHz	1000,000000Hz
Rechteck 10kHz	10000,000000Hz

Tabelle 4.2: Messergebnisse für obertonhaltige Signalformen bei Nutzung der FFT-Methode bei einer Abtastrate von 48kHz und einer Fensterbreite von 16384 Samples

4.4 Ergebisse: McLeod-Algorithmus

Der McLeod-Algorithmus zeigte eine sehr gute Erkennung des gespielten Tones auch bei stark verrauschten Signalen. Die Genauigkeit des Algorithmus nimmt zu hohen Frequenzen hin ab. Dies ist durch die in der Interpolation begründete Abweichung verursacht. Ab ca. $^1/_5$ der Abtastfrequenz nehmen die Messabweichungen zu. Ursächlich hierfür ist die Parabelinterpolation, deren Genauigkeit stark abnimmt, wenn Wendepunkte zwischen den Stützstellen liegen.

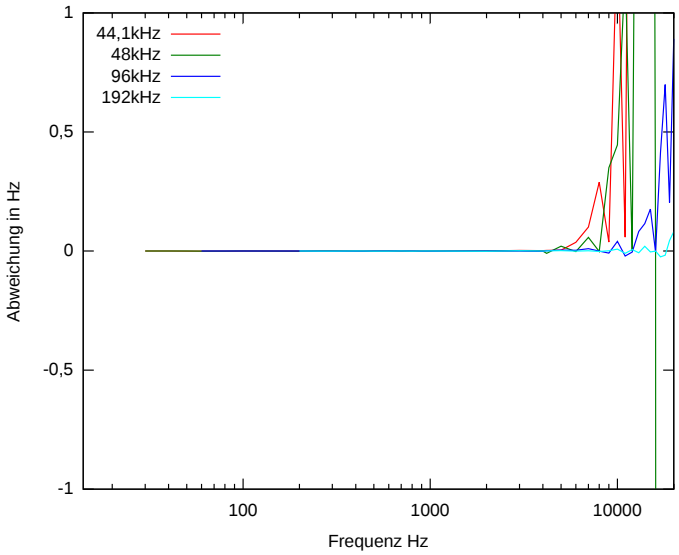


Abbildung 4.6: Messgenauigkeit des McLeod-Algorithmus mit Parabelinterpolation bei 4096 Samples Fensterbreite bei unterschiedlichen Abtastraten

Bei der Auswertung der Messergebnisse zeigte sich, dass die Abtastrate über einen weiten Teil des relevanten Spektrums keinen nennenswerten Einfluss auf die Genauigkeit hat, jedoch entscheidend für die obere Grenzfrequenz ist, bis zu der die Interpolation zuverlässig genaue Ergebnisse erzielt. Dargestellt ist dies zum Beispiel in Abbildung 4.6.

Die Fensterbreite bestimmt gemeinsam mit der Abtastrate maßgeblich die untere Grenzfrequenz, bis zu der der Algorithmus zuverlässig arbeitet. Im Beobachtungsfenster sollten die Datenwerte von wenigstens 2 Signalperioden gespeichert sein, ist das Fenster kleiner, so ist eine Erkennung der Signalfrequenz nicht möglich. Wie in Abbildung 4.7 ist auch ein geringer Einfluss auf die Genauigkeit der Messergebnisse zu verzeichnen.

Wie aus Abbildung 4.8 erkennbar ist, kann der McLeod-Algorithmus mithilfe der Brent-Methode über den gesamten Frequenzbereich eingesetzt werden, jedoch ist die Messgenauigkeit geringer als bei Nutzung der Parabelinterpolation.

Die Messung alternativer Signalformen ergab, dass bei starker Obertonhaltigkeit die Messgenauigkeit in geringem Umfang abnimmt. Durch eine zusätzliche Filterung des

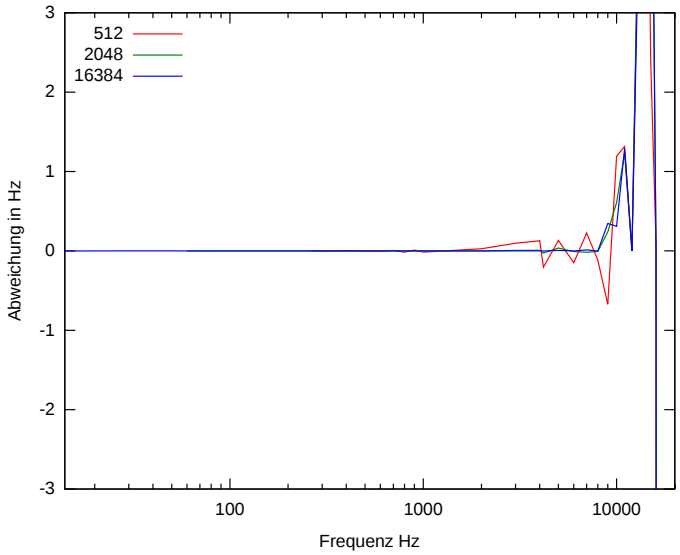


Abbildung 4.7: Messgenauigkeit des McLeod-Algorithmus mit Parabelinterpolation bei verschiedenen Fensterbreiten und 48kHz Abtastrate

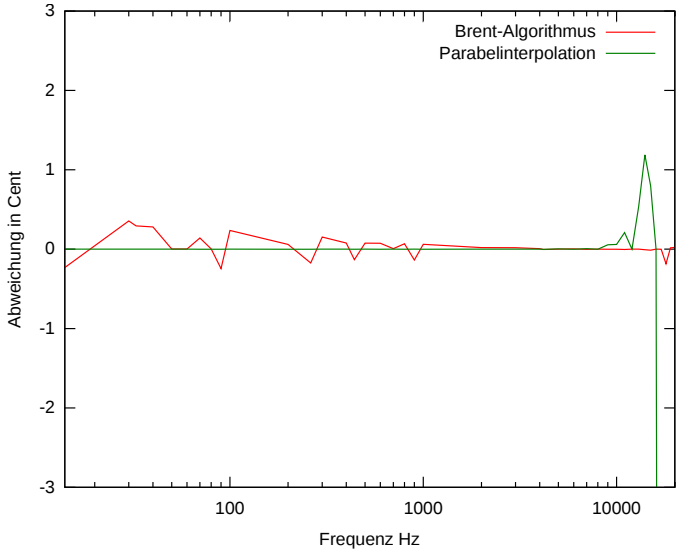


Abbildung 4.8: Relativer Messfehler des McLeod-Algorithmus bei Interpolation mit Parabel- und Brentalgorithmus, Abtastrate 48kHz über 8192 Samples

Frequenz	McLeod Parabelinterpolation	McLeod Brentmethode
Sägezahn 14Hz	14,000009Hz	13,998184Hz
Sägezahn 100Hz	100,000007Hz	100,013585Hz
Sägezahn 1kHz	999,999944Hz	1000,031624Hz
Sägezahn 10kHz	10000,348061Hz	9999,994435Hz
Rechteck 14Hz	14,041219Hz	14,039397Hz
Rechteck 100Hz	100,000007Hz	100,001240Hz
Rechteck 1kHz	1000,000068Hz	1000,004150Hz
Rechteck 10kHz	10000,348058Hz	9999,994432Hz

Tabelle 4.3: Messergebnisse für obertonhaltige Signalformen bei Nutzung des McLeod-Algorithmus bei einer Abtastrate von 48kHz und einer Fensterbreite von 8192 Samples

Signales um die Sollfrequenz könnte dieser Messfehler vermindert werden.

Die Auswertung der Orgelpfeifenaufnahmen zeigte, dass der Algorithmus bei vielen dominanten Obertönen dazu neigt, diese als Grundton zu erkennen. Laute Störtöne fester Frequenz verfälschen jedoch das Messergebnis stark und müssen vermieden werden.

5 Fazit und Ausblick

Die Untersuchungsergebnisse lassen die Implementierung eines Stimmgerätes in ein Embedded Linux System möglich erscheinen. Die Standardabtastraten 44,1kHz; 48kHz und 96kHz haben sich als guter Kompromiss zwischen der Verringerung der Anzahl nötiger Abtastwerte und der Steigerung der Messgenauigkeit erwiesen. Kleinere Abtastraten verletzten die Nyquist-Bedingung, wohingegen größere den Rechenaufwand signifikant erhöhten, jedoch keine für das Projekt relevante Senkung der Messfehler bewirkten. Der notwendige Dynamikumfang der Signale erwies sich als unkritisch. Auch bei Aufzeichnungen von Orgeltönen mit sehr geringer Aussteuerung und einem Dynamikumfang von weniger als 55 dB konnte die Frequenz sicher bestimmt werden. Soundkarten mit einer 16 Bit AD-Wandlung genügen diesen Anforderungen mit ihrem Dynamikumfang von 96 dB. Um den verfügbaren Dynamikumfang der Soundkarte möglichst voll auszuschöpfen und so das bestmögliche Signal-zu-Rauschverhältnis zu erzielen, sollte die Mikrofonvorverstärkung einstellbar sein. Auf diese Weise könnte sowohl bei leisen als auch bei lauten Orgelpfeifen eine Aussteuerung nahe 0 db_{fs}[5] realisiert werden.

Als besonders genau hat sich der FFT-Algorithmus ab Frequenzen oberhalb von 1000Hz erwiesen, die Grenze, ab der dieser Algorithmus die besten Ergebnisse bringt, ist abhängig von Abtastrate und Fensterbreite.

Der zweite Algorithmus, der sehr gute Messergebnisse lieferte, war der McLeod-Algorithmus. Die besten Ergebnisse konnten hier bei einer Abtastrate von 96kHz und 16384 verarbeiteten Samples erzielt werden. Auch bei einer Abtastrate von 48kHz und einer Fensterbreite von 8192 Samples würde dieser Algorithmus den Anforderungen entsprechen.

Die Implementierung des Autokorrelationsalgorithmus zeigte hingegen über weite Bereiche des Spektrums keine ausreichend genauen Messergebnisse.

Als geeigneter Kompromiss kann folgende Lösung für den Frequenzbereich bis 500Hz dienen:

- Abtastrate: 48kHz
- Fensterbreite: 8192 Samples
- Algorithmus: McLeod
- Interpolation: Parabelinterpolation

Ab 500Hz kann die gesteigerte Genauigkeit des FFT-Algorithmus genutzt werden:

- Abtastrate: 48kHz

[5] dB bezogen auf die Vollaussteuerung der Soundkarte (full scale), Positive Werte bedeuten eine Übersteuerung des Eingangs.

- Fensterbreite: 4096 Samples

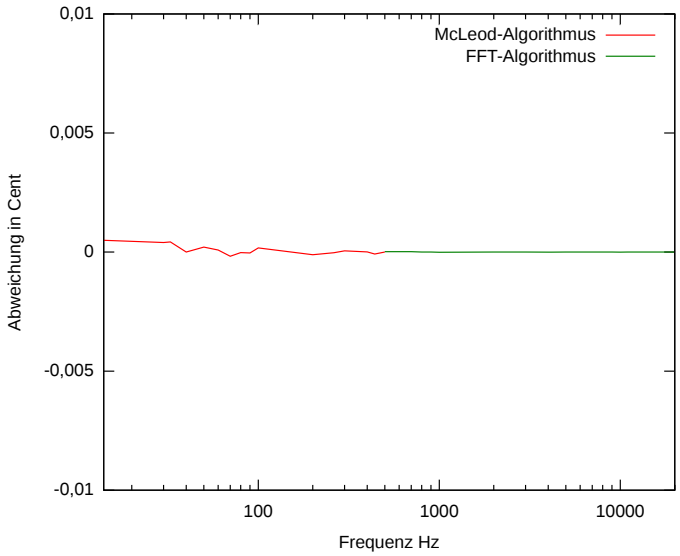


Abbildung 5.1: Messgenauigkeit der Kombination aus FFT und McLeod-Algorithmus

Eine weitere Möglichkeit wäre die Verwendung des McLeod-Algorithmus bei einer Abtastrate von 96kHz, einer Fensterbreite von 16384 Samples und unter Nutzung der Parabelinterpolation. Obwohl diese Konfiguration weniger genau als die zuvor vorgeschlagene ist, wird sie den Anforderungen an die Genauigkeit gerecht. Es ist jedoch mit einer höheren Verarbeitungszeit zu rechnen.

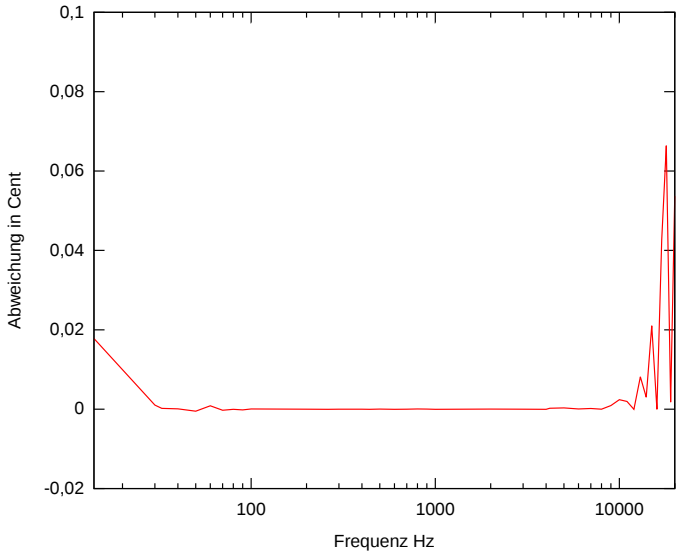


Abbildung 5.2: Messgenauigkeit McLeod-Algorithmus bei einer Abtastrate von 96kHz und einer Fensterbreite von 16384 Samples

Um die Fehlerkennung von Tönen zu vermeiden und den Einfluss von Störungen auf das Messergebnis zu senken, könnte, ähnlich wie im Stimmgerät TLA CTS-32-C implementiert, ein steilflankiger Filter, dessen Mittenfrequenz der Sollfrequenz des gespielten

Tones entspricht, auf das Signal angewendet werden. In der Folge würden die Obertöne des Signales unterdrückt, so dass der FFT-Algorithmus diese nicht mehr erkennen würde. Des Weiteren würden Störgeräusche zum Beispiel das Betriebsgeräusch des Orgelmotors das Messergebnis weniger stark beeinflussen.

Weiterhin ist zu prüfen, inwieweit der McLeod-Algorithmus durch Patente geschützt ist. Sollte eine Nutzung des Algorithmus nicht möglich sein, könnte eine Abtastratenreduktion mit Tiefpassfilterung des Signales eine Lösung sein, auch mit dem FFT-Algorithmus im Bereich von 14Hz bis 500Hz hinreichend genaue Ergebnisse zu erzielen.

Aufgrund der Möglichkeit, dass ein Pfeifenton tremuliert müssen mit einigem zeitlichen Abstand einige wenige Messwerte aufgenommen werden. Dies kann durch mehrfaches Aufrufen des Messalgorithmus realisiert werden. Weisen die ermittelten Werte eine große Varianz auf, so sollte die Anzahl der Messewerte erhöht werden, um anschließend eine Mittelwertbildung vorzunehmen. Auf diese Weise kann die Mittenfrequenz, um die der Ton tremuliert, bestimmt werden.

Literaturverzeichnis

[1] HOCHSCHULE MITTWEIDA - LABOR EMBEDDED CONTROL: Sachbericht zum Thema: Pfeifen-Orgel mit dynamischer Stimmung. Mittweida, Dezember 2013. – Forschungsbericht

[2] HAUNSCHILD, Frank: *Die neue Harmonielehre Bd. 1*. AMA Verlag, 1998. – ISBN 3–927190–00–4

[3] Norm DIN 13320 (1979). *Akustik; Spektren und Übertragungskurven, Begriffe, Darstellung*

[4] PHILLIP, Kuhrt: *Piano keyboard with notes and names*. `http://commons.wikimedia.org/wiki/File:KlaviaturMitNoten.svg`, Abruf: 06.02.2014, 10:45

[5] Norm DIN 1317 Teil 3 (1962). *Stimmton der Orgel*

[6] ADELUNG, Wolfgang: *Einführung in den Orgelbau*. - 6. Aufl. - Leipzig : VEB Breitkopf & Härtel Musikverlag, 1972

[7] SCHEITHAUER, Rainer: *Signale und Systeme*. Stuttgart : Teubner, 1998 (Leitfaden der Elektrotechnik). – ISBN 3–519–06425–1

[8] MCLEOD, Philip ; WYVILL, Geoff: *A smarter way to find pitch*. Proc. International Computer Music Conference, 2005

[9] KAMMEYER, Karl-Dirk ; KROSCHEL, Kristian: *Digitale Signalverarbeitung: Filterung und Spektralanalyse*. - 4. Aufl. - Stuttgart : Teubner, 1998. – ISBN 3–519–36122–1

[10] ZÖLZER, Udo: *Digitale Audiosignalverarbeitung*. - 3. Aufl. - Vieweg+Teubner Verlag, 2005 (Teubner Informationstechnik). – ISBN 978–3–519–26180–3

[11] GERHARD, David: *Pitch Extraction and Fundamental Frequency: History and Current Techniques*. 2003. – ISBN 0–7731–0455–0

[12] BOERSMA, Paul: Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In: *IFA Proceedings*

[13] PRESS, William ; FLANNERY, Brian ; TEUKOLSKY, Saul ; VETTERLING, William: *Numerical Recipes in C: The Art of Scientific Computing*. - 2nd ed. -. Cambridge University Press, 1992. – ISBN 0521431085

Anhang A: Quellcode der Algorithmen

Im Anhang werden die zur Untersuchung verwendeten Quelltexte gezeigt. Zur besseren Darstellung und Übersichtlichkeit wurden die Texte wie folgt Modifiziert:

- Entfernung von Debugausgaben
- Entfernung von Include-Anweisungen
- Entfernung von Funktionsprototypen
- Anpassung von Zeilenumbrüchen
- Ergänzung von Definitionen aus Headerdateien
- Verzicht auf Unterfunktionen

Die exakte Implementierung ist Anhang B zu entnehmen.

A.1 Autokorrelationsalgorithmus

Listing A.1: Autokorrelationsalgorithmus

```
1  /**
2   * @brief Frequenzschaetzung nach dem Praat AC Algorithmus
3   * @param data   Zeiger auf das Feld mit den Samples
4   * @param samplerate   Abtastrate der Daten
5   * @param N   Anzahl zu verwendender Samples
6   * @param window_fkt   0   Gaus Fenster; 1   Kaiser Fenster
7   * @param brent   0 Parabelschaetzung; 1 Brent Algorithmus
8   */
9  double praat_ac(double *data, int samplerate,
10                 int N, int window_fkt, int use_brent)
11 {
12     double data_max=0;
13     double data_avg=0;
14     double data_in[N];
15     double korrelation[N];
16     double window[N];
17     double akf_window[N];
18     double erg[N];
19     int i;
20
21     memcpy(data_in, data, N*sizeof(double));
22     for(i=0; i<N; i++)
23     {
24         data_max = (data_in[i] > data_max ? data_in[i] : data_max);
25         data_avg += data_in[i];
```

```
26      }
27      data_avg /= N;
28      for(i=0; i<N; i++)
29      {
30          data_in[i] = data_avg;
31      }
32      //Berechne Fensterfunktion & bilde die AKF
33      if(window_fkt == 1)
34          kaiser_window(N, 0.5, window);
35      else
36          gaus_window(N, 0.5, window);
37      akf(window, N, akf_window);
38
39      use_window(data_in, N, window); //Fenstere die Eingangsdaten
40      akf(data_in, N, korrelation); //Bilde die AKF der Daten
41
42      for(i=1; i<N; i++) //Normiere die AKF der Fensterfunktion
43      {
44          akf_window[i] /= akf_window[0];
45          korrelation[i] /= korrelation[0];
46          erg[i] = korrelation[i] / akf_window[i];
47      }
48      erg[0] = korrelation[0] / akf_window[0];
49
50
51      double freq=0;
52      double max=0;
53      double last_max = 0;
54      int count_max = 0;
55      for(i=2; i<(N 1); i++)
56      {
57          if( erg[i]>erg[i+1] &&
58              erg[i]>erg[i 1] )
59          {
60              double act_freq=0, ymax;
61              max = i;
62              double umg[3] = { max 1 , max, max+1};
63              if(use_brent == 1)
64                  ymax = brent( umg[0], umg[1], umg[2], 10e 6, &act_freq, N, erg , 0);
65              else
66                  act_freq = interpolate_max_3_y(umg, &erg[i 1], &ymax);
67              if(ymax > SCHWELLWERT || erg[i] > SCHWELLWERT)
68              {
69                  freq += act_freq last_max;
70                  last_max = act_freq;
71                  count_max++;
```

```
72                  }
73              }
74          }
75          if(max == 0)    //kein Maximum gefunden
76          {
77              return 0;
78          }
79          freq /= (double)count_max;
80          return samplerate / freq;
81      }
```

A.2 FFT-Algorithmus

Listing A.2: FFT-Algorithmus

```
1  /**
2   * @brief Funktion zur Ermittlung des staerksten Partialtones eines Signals
3   * @param *data    Zeiger auf das Datenfeld mit den Eingangswerten
4   *                 mit der mindestgroesse N+2
5   * @param samplerate    Abtastrate des Signals
6   * @param N    Aufloesung der FFT
7   * @return Frequenz des staerksten Partialtones
8   */
9
10 double fft_dif(double *data, int samplerate, int N)
11 {
12     int k = 0;
13     int max1=0;
14     fftw_complex *out1, *out2;
15     fftw_plan p1, p2;
16     double erg[N/2];
17     double grund_freq=0;
18     double data_set1[N], data_set2[N];
19     double window[N];
20
21     //Kopiere die Daten
22     memcpy(data_set1, data, N*sizeof(double));
23     memcpy(data_set2, data+2, N*sizeof(double));
24     //Alloziere Speicher fuer Ergebnisse der FFT
25     out1 = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
26     out2 = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
27
28     //Erstelle die Fensterfunktion
29     kaiser_window(N, 8, window);
30     //Wende das Fenster auf beide Datenfelder an
31     use_window(data_set1, N, window);
32     use_window(data_set2, N, window);
33     //Fuehre die FFT aus
34     p1 = fftw_plan_dft_r2c_1d(N, data_set1, out1, FFTW_ESTIMATE);
35     p2 = fftw_plan_dft_r2c_1d(N, data_set2, out2, FFTW_ESTIMATE);
36     fftw_execute(p1);
37     fftw_execute(p2);
38
39     //Bilde den Betrag und bestimme das Maximum
40     for(k=0; k<(N/2); k++)
41     {
42         erg[k] = cabs(out1[k]);
43     }
```

```
44      for(k=0; k<(N/2); k++)
45      {
46          if(erg[k] > erg[max1])
47              max1=k;
48      }
49      //Berechne das Argument der Maxima
50      double phi1 = carg(out1[max1]);
51      double phi2 = carg(out2[max1]);
52      //und bilde die Differenz
53      double winkel = phi2 phi1;
54      if(winkel < 0) winkel += 2*PI;
55      //Berechne die Frequenz aufgrund der Winkelaenderung
56      //waehrend der Verschiebungszeit
57      grund_freq = samplerate * ((winkel)/(2*2*PI));
58      //Gib allozierten Speicher frei
59      fftw_destroy_plan(p1);
60      fftw_destroy_plan(p2);
61      fftw_free(out1);
62      fftw_free(out2);
63      return grund_freq;
64  }
```

A.3 McLeod-Algorithmus

Listing A.3: McLeod-Algorithmus

```
1  #define TOLL_MCLEOD 0.8
2
3  typedef struct max_pos {
4      int x;
5          struct max_pos *next;
6          }max_pos;
7  double mcleod(double *data, int samplerate, int m, int brent);
8
9  /**
10  * @brief Pitchdetection mithilfe des McLeod Algorithmus
11  *
12  * @param *data   Zeiger auf das Datenfeld
13  * @param samplerate   Abtastfrequenz des Signals
14  * @param m Groesse des Datenfelds
15  * @param brent   verwenden des Brentalgoritmus zur maximasuche? 0: Nein; sonst: Ja
16  * @return Ermittelte Grundfrequenz
17  */
18 double mcleod(double *data, int samplerate, int m, int use_brent)
19 {
20     int i = 0 ,j = 0;
21     int r = 0;
22
23     double erg[m];
24     double rt, mt;
25     for(r=0; r<m; r++)    //Berechne die NSDF
26     {
27         erg[r]=0;
28         rt = 0;
29         mt = 0;
30         for(j=0; j<(m r 1); j++)
31         {
32             rt += data[j]*data[j+r];
33             mt += (data[j]*data[j]) + (data[j+r]*data[j+r]);
34         }
35         erg[r] = 2*rt / mt;
36     }
37     erg[m1]=0;
38
39     //Finde relevante Maximalstellen und den Maximalen Wert
40     max_pos *maxima=NULL;
41     max_pos *maxima_ptr=maxima;
42     max_pos *tmp_pos=maxima;
43     double biggest_max=0, schwellwert=TOLL_MCLEOD;
```

```
44      for(i=1; i<(m-1); i++)
45      {
46          if(erg[i-1]<0 && erg[i]>=0) //suche Nulldurchgang, positive Flanke
47          {
48              if(maxima_ptr!=NULL)    //der Zeiger ist gueltig
49              {
50                  biggest_max = (biggest_max < erg[maxima_ptr->x]) ?
51                  erg[maxima_ptr->x] : biggest_max; //bestimme groesstes Maximum
52                  maxima_ptr->next = malloc(sizeof(max_pos));
53                  maxima_ptr = maxima_ptr->next;
54                  maxima_ptr->next = NULL;
55                  maxima_ptr->x = i;
56              }
57              else
58              {
59                  maxima_ptr = malloc(sizeof(max_pos));
60                  maxima = maxima_ptr;
61                  maxima_ptr->next = NULL;
62                  maxima_ptr->x = i;
63              }
64          }
65          if(maxima_ptr!=NULL) //Finde Maximum
66          {
67                if(erg[i]>=erg[maxima_ptr->x] && erg[i]>=erg[i+1])
68                {
69                    maxima_ptr->x = i;
70                }
71          }
72      }
73
74      //Berechne genaue Maximalpunkte und gib den Speicher wieder frei
75      maxima_ptr = maxima;
76      schwellwert = TOLL_MCLEOD * biggest_max;
77      double last_max = 0;
78      double mittelwert=0;
79      j = 0;
80      while(maxima!=NULL)
81      {
82          if( (maxima->x) > 50 && (maxima->x) < (m*0.9) && j<5000)
83          {
84              double act_max, ymax;
85              if(use_brent != 0)
86              {
87                  double xmin;
88                  ymax = brent( (maxima->x)-1,
89                                maxima->x,
```

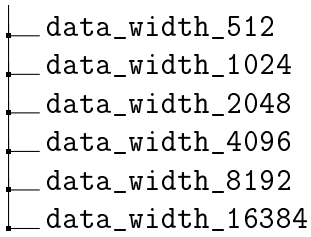
```
90                          (maxima->x)+1  ,
91                          1e-7, &xmin, m, erg, 0);
92                  act_max = xmin;
93              }
94              else
95              {
96                  double umg[3];
97                  umg[0] = (maxima->x) - 1;
98                  umg[1] = maxima->x;
99                  umg[2] = (maxima->x)+1;
100                 act_max = interpolate_max_3_y(umg,
101                                 &erg[(maxima->x) - 1],
102                                 &ymax);
103             }
104             if(last_max != 0 &&
105               (ymax > schwellwert || erg[maxima->x] > schwellwert))
106             {
107                 mittelwert += (double)samplerate/(act_max - last_max);
108                 j++;
109             }
110             last_max = act_max;
111         }
112         tmp_pos = maxima->next;
113         free(maxima);
114         maxima = tmp_pos;
115     }
116     return mittelwert/(double)j;
117 }
```

Anhang B: Datenträger

Dateistruktur der beiligenden DVD:

```
├── Abschlussarbeit
│   └── Abschlussarbeit.pdf ... Die vorliegende Arbeit als PDF-Datei.
├── Applikation ... Enthält die Quelltexte der verwendeten
│                   Applikationen (siehe Abschnitt 3.7).
│   ├── analyser
│   ├── fourier_printer
│   ├── hz2cent
│   ├── lib
│   ├── wave_gen
│   ├── wave_gen_cmd
│   ├── gen_script
│   └── scan_script
├── Messung_Orgel ... Enthält die Aufnahmen und Messergebnisse der
│                     untersuchten Orgel.
│   ├── cornet_2 ... Aufzeichnungen: Register Cornet-2′
│   ├── gedackt_8 ... Aufzeichnungen: Register Gedackt-8′
│   ├── oboe_8 ... Aufzeichnungen: Register Oboe-8′
│   ├── prinzipal_8 ... Aufzeichnungen: Register Prinzipal-8′
│   ├── salicional_8 ... Aufzeichnungen: Register Salicional-8′
│   ├── spitzoktave_4 ... Aufzeichnungen: Register Spitzoktave-4′
│   ├── subbass_16 ... Aufzeichnungen: Register Subbass-16′
│   └── waldfloete_2 ... Aufzeichnungen: Register Waldflöte-2′
├── Messung_Rechteck ... Enthält Aufnahmen und Messergebnisse der
│                        Messung mit Rechtecksignalen.
├── Messung_Saegezahn ... Enthält Aufnahmen und Messergebnisse der
│                         Messung mit Sägezahnförmigen Signalen.
└── Messung_Sinus ... Enthält Aufnahmen und Messergebnisse der
    │                 Messung mit Sinussignalen.
    ├── SR44100 ... Aufzeichnungen mit Abtastrate 44,1kHz
    ├── SR48000 ... Aufzeichnungen mit Abtastrate 48kHz
    ├── SR96000 ... Aufzeichnungen mit Abtastrate 96kHz
    └── SR192000 ... Aufzeichnungen mit Abtastrate 192kHz
```

Alle Verzeichnisse mit Audioaufzeichnungen (wave-Dateien) enthalten weiterhin Verzeichnisse:



Diese enthalten die Messergebnisse aller Algorithmen und die zugehörigen Fastfourier-Transformationen, Autokorrelierten und der normierten quadratischen Differenzgleichungen. Die Textdateien *M_absolut.txt*, *M_cent.txt*, *M_relativ.txt* und *Messung.txt* enthalten die Messergebnisse in tabellarischer Form.

- *M_absolut.txt* - Absolute Messergebnisse $f - f_{soll}$
- *M_cent.txt* - Abweichung in Cent $1200 * \log_2 \left(\frac{f}{f_{soll}} \right)$
- *M_relativ.txt* - Relative Abweichung $\frac{f}{f_{soll}}$
- *Messung.txt* - Messergebnisse in Hertz

In, durch Tabulatoren getrennten, Spalten stehen dabei der Reihe nach folgende Werte:

1. Die Sollfrequenz f_{soll}
2. Das Messergebnis des Autokorrelationsalgorithmus mit der Brent-Methode
3. Das Messergebnis des Autokorrelationsalgorithmus mit Parabelinterpolation
4. Das Messergebnis des McLeod-Algorithmus mit der Brent-Methode
5. Das Messergebnis des McLeod-Algorithmus mit Parabelinterpolation
6. Das Messergebnis der FFT-Methode

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.
Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 14.02.2014