TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

Juniorprofessur Media Computing

Masterarbeit

Entwicklung, Optimierung und Evaluation eines Algorithmus zur Erkennung akustischer Schwebungen bei gleichzeitig erklingenden Labial- und Lingualpfeifen von Orgeln

Nico Junghanns

Chemnitz, 1. November 2022

Prüfer: Jun.-Prof. Dr. rer. nat. Danny Kowerko
Betreuer: M.Sc. René Erler

Danksagung

Ich möchte mich bei allen bedanken, die mich bei der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Mein besonderer Dank gilt der Juniorprofessur Media Computing für die Möglichkeit, dieses Thema als Abschlussarbeit umsetzen zu können.

Ebenso bedanke ich mich bei der Ev.-Luth. Kirchgemeinde St. Petri Augustusburg für die Möglichkeit Fotoaufnahmen der Jehmlich-Orgel zu machen.

Abschließend möchte ich mich bei meinen Eltern und meiner gesamten Familie bedanken, die mich während meines Studiums unterstützt haben.

Zusammenfassung

Bei der Überlagerung akustischer Schwingungen mit geringem Frequenzunterschied kommt es zu einem periodischen An- und Abschwellen der Lautstärke. Dieses Phänomen wird als akustische Schwebungen bezeichnet. Deren Frequenz, also die Anzahl der auftretenden Lautstärkeminima bzw. -maxima pro Sekunde, entspricht exakt dem Abstand der beiden Ausgangsfrequenzen.

Schwebungen werden vor allem bei der Intonation und dem Stimmen verschiedener Musikinstrumente, wie z.B. Klavieren, Gitarren oder auch Orgeln, eingesetzt. Bei Letzteren kommt es regelmäßig durch Temperaturänderungen zu Verschiebungen der Frequenzen der verschiedenen Pfeifenarten, hauptsächlich Labial- und Lingualpfeifen, und somit zur Inharmonizität zwischen diesen. Das macht ein regelmäßiges Nachstimmen notwendig. Dies erfolgt unter Zuhilfenahme und Elimination der auftretenden Schwebungen.

Aktuell existieren keine Algorithmen, um solche Schwebungen bei verschiedenartigen Orgelpfeifen zu erkennen. Im Rahmen dieser Masterarbeit wurde ein solcher Algorithmus entwickelt und evaluiert. Dabei wurden verschiedene Ansätze untersucht und miteinander verglichen. Insgesamt konnten mit dem Algorithmus bis zu 85 % aller Schwebungen in den Testdaten korrekt erkannt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis iv

Tabellenverzeichnis vii

Algorithmenverzeichnis viii

Abkürzungsverzeichnis ix

1 Einleitung 1
- 1.1 Kontext . . . 1
- 1.2 Ziele der Arbeit und Hypothesen . . . 1
- 1.3 Gliederung der Arbeit . . . 2

2 Technischer Hintergrund 3
- 2.1 Musiktheoretische Grundlagen . . . 3
  - 2.1.1 Aufbau der Klaviatur . . . 3
  - 2.1.2 Diatonische Tonsysteme und Intervalle . . . 4
  - 2.1.3 Stimmungssysteme . . . 5
- 2.2 Orgeln . . . 7
  - 2.2.1 Aufbau . . . 7
  - 2.2.2 Funktionsweise . . . 9
  - 2.2.3 Labialpfeifen . . . 10
  - 2.2.4 Lingualpfeifen . . . 11
  - 2.2.5 Vorgehen beim Stimmen . . . 12
- 2.3 Schwebungen . . . 13

3 Stand der Technik 16
- 3.1 Existierende Stimmgeräte . . . 16
- 3.2 Monophone Frequenzerkennung . . . 16
  - 3.2.1 Algorithmen im Zeitbereich . . . 18
  - 3.2.2 Algorithmen im Frequenzbereich . . . 20
- 3.3 Polyphone Frequenzerkennung . . . 26

3.4 Zusammenfassung 27

4 Implementierung **28**
4.1 Überblick 28
4.2 Frequenzerkennung 29
4.3 Schwebungserkennung 30
4.3.1 Analyse der Amplituden 30
4.3.2 Analyse per Hüllkurvenerkennung 32
4.3.3 Analyse der Fourierkoeffizienten 34
4.3.4 Grundlegende Überprüfung auf das Vorhandensein einer Schwebung 37
4.4 Implementierung der verwendeten Algorithmen 38
4.5 Softwarearchitektur 38
4.6 Testumgebung 40
4.7 Programmierumgebung 41

5 Generierung der Testdaten **43**
5.1 Aufbau der Testdaten 43
5.2 Erzeugung der Testdaten 46
5.2.1 Allgemeines Vorgehen 46
5.2.2 Additive Synthese 48
5.2.3 GrandOrgue-basierte Synthese 49
5.3 Überblick über die verwendeten Testsets 50

6 Evaluation **51**
6.1 Frequenzerkennung 51
6.1.1 Methodik 51
6.1.2 Ergebnisse für additiven Testdaten 53
6.1.3 Ergebnisse für GrandOrgue-basierte Testdaten 56
6.2 Schwebungserkennung 59
6.2.1 Methodik 59
6.2.2 Ergebnisse für additive Testdaten 61
6.2.3 Ergebnisse für GrandOrgue-basierte Testdaten 67
6.3 Gesamtsystem 69
6.3.1 Methodik 69
6.3.2 Ergebnisse für additive Testdaten 70
6.3.3 Ergebnisse für GrandOrgue-basierte Testdaten 71
6.4 Vergleich der verschiedenen Pitch-Shifting-Algorithmen 72
6.4.1 Methodik 72

6.4.2 Ergebnisse für additive Testdaten 73
6.4.3 Ergebnisse für GrandOrgue-basierte Testdaten 74

7 Fazit und Ausblick **76**
7.1 Fazit 76
7.2 Ausblick 77

Literaturverzeichnis **79**

A Anhang **83**
A.1 Implementierung der verwendeten Algorithmen 83
A.1.1 Clustering-Algorithmus 83
A.1.2 Interpolationsalgorithmus für FFT-basierte Ergebnisse 87
A.1.3 Frequenzerkennungsalgorithmen 91
A.2 Parametrierung der evaluierten Algorithmen 103
A.2.1 Algorithmen für die Frequenzerkennung 103
A.2.2 Algorithen für die Schwebungserkennung 104

Abbildungsverzeichnis

2.1 Schematischer Aufbau einer Klaviatur 3
2.2 Vergleich der Frequenzverhältnisse von natürlicher und gleichstufiger Stimmung 7
2.3 Spieltisch einer Orgel 8
2.4 Aufbau einer Schleifladenorgel 9
2.5 Aufbau einer Labialpfeife 10
2.6 Aufbau einer Lingualpfeife 12
2.7 Stimmeisen 13
2.8 Entstehung von Schwebungen 13

3.1 Benutzeroberfläche von Dirk's Stimmgerät für chromatische Instrumente . 17
3.2 Vorgehen zur Ermittlung der ZCR 18
3.3 Berechnung von ACF und SDF 19
3.4 Frequenzerkennung per FFT 21
3.5 Frequenzerkennung per HPS 22
3.6 Frequenzerkennung per HSA 23
3.7 Frequenzerkennung per Cepstrum 24

4.1 Datenflussdiagramm des entwickelten Schwebungserkennungsalgorithmus . 28
4.2 Datenflussdiagramm des Frequenzerkennungsalgorithmus 29
4.3 Datenflussdiagramm für die Schwebungserkennung durch Analyse der Amplituden 31
4.4 Vorgehen zur Ermittlung der Hüllkurve durch Analyse der Amplituden des Signals 32
4.5 Datenflussdiagramm für die Schwebungserkennung durch Analyse per Hüllkurvenerkennung 33
4.6 Datenflussdiagramm für die Hüllkurvenerkennung 33
4.7 Vorgehen zur Ermittlung der Hüllkurve des Signals per Hüllkurvenerkennung 34
4.8 Verläufe von Fourierkoeffizienten von Grund- und Obertönen für ein Signal mit vorliegender Schwebung 35

4.9 Datenflussdiagramm für die Schwebungserkennung durch Analyse der Fourierkoeffizienten . . . 36
4.10 Klassendiagramm der Python-Implementierung des Gesamtsystems . . . 39
4.11 Aufbau eines Testsets . . . 40

5.1 Bureå Church (extended AGO version)-Orgel in GrandOrgue . . . 43
5.2 Die Überlagerung von zwei Schwingungen mit den Frequenzen 120 Hz (oben) und 32 Hz (Mitte) führt zu keiner sichtbaren Schwebung (unten) . 45
5.3 Datenflussdiagramm für die Erzeugung der Testdaten aus Audioaufnahmen von zwei Orgelregistern . . . 47
5.4 Datenflussdiagramm zum Mischen der Testdaten aus zwei einzelnen Tönen 48

6.1 Veranschaulichung der maximal erlaubten Abweichung für korrekt erkannte Frequenzen . . . 52
6.2 Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Frequenzerkennungsverfahren mit additiven Testdaten . . . 55
6.3 Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Frequenzen für die verschiedenen Frequenzerkennungsverfahren mit additiven Testdaten . . . 56
6.4 Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Frequenzerkennungsverfahren mit GrandOrgue-basierten Testdaten . . . 57
6.5 Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Frequenzen für die verschiedenen Frequenzerkennungsverfahren mit GrandOrgue-basierten Testdaten . . . 58
6.6 Veranschaulichung der maximal erlaubten Abweichung für korrekt erkannte Schwebungsfrequenzen . . . 60
6.7 Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Amplitude mit additiven Testdaten . . . 62
6.8 Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Hüllkurve mit additiven Testdaten . . . 64
6.9 Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit additiven Testdaten . . . 65
6.10 Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen für die verschiedenen Verfahren zur Schwebungserkennung mit additiven Testdaten . . . 66

6.11 Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit GrandOrgue-basierten Testdaten 68
6.12 Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen für die verschiedenen Verfahren zur Schwebungserkennung mit GrandOrgue-basierten Testdaten 69
6.13 Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit additiven Testdaten 71
6.14 Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit GrandOrgue-basierten Testdaten 72
6.15 Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit per pyrubberband erstellten additiven Testdaten 73
6.16 Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit per pyrubberband erstellten GrandOrgue-basierten Testdaten 74

A.1 Beispielhafter Ablauf des Clustering-Algorithmus mit sieben Clustern . . . 84
A.2 Berechnung des FFT-Maximums per Interpolation 89

Tabellenverzeichnis

2.1 Überblick über Intervalle 4
2.2 Frequenzverhältnisse in der natürlichen Stimmung 5

5.1 Übersicht über die vorhandenen Labial- und Lingualregistern der verwendeten Bureå Church (extended AGO version)-Orgel 44
5.2 Überblick über verwendete Testsets 50

6.1 Einteilung der Frequenzbereiche für die Testfälle der Frequenzerkennung . 54
6.2 Einteilung der Frequenzbereiche für die Testfälle der Schwebungserkennung 61

A.1 Parametrierungen der Frequenzerkennungsverfahren 103
A.2 Parametrierungen der Schwebungserkennungsverfahren per Analyse der Amplituden mit additiven Testdaten 104
A.3 Parametrierungen der Schwebungserkennungsverfahren per Analyse der Hüllkurve mit additiven Testdaten 105
A.4 Parametrierungen der Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit additiven Testdaten 106
A.5 Parametrierungen der Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit GrandOrgue-basierten Testdaten 107

Algorithmenverzeichnis

A.1 Clustering-Algorithmus 85
A.2 Algorithmus zum Initialisieren der Cluster 86
A.3 Algorithmus zum Zusammenfassen aller Cluster 87
A.4 Algorithmus zum Zusammenfassen von zwei Clustern 88
A.5 Algorithmus zur Interpolation von FFT-Ergebnissen 90
A.6 Algorithmus zur Frequenzerkennung per ZCR 91
A.7 Algorithmus zur Initialisierung der ZCR-Berechnung 92
A.8 Algorithmus zur Frequenzerkennung per ACF 94
A.9 Algorithmus zum Finden der lokalen ACF-Maxima 95
A.10 Algorithmus zur Frequenzerkennung per SDF 96
A.11 Algorithmus zum Finden der lokalen SDF-Minima 97
A.12 Algorithmus zur Frequenzerkennung per FFT 97
A.13 Algorithmus zur Frequenzerkennung per HPS 99
A.14 Algorithmus zur Frequenzerkennung per HSA 100
A.15 Algorithmus zum Finden der FFT-Maxima 101
A.16 Algorithmus zur Frequenzerkennung per Cepstrum 102

Abkürzungsverzeichnis

ACF Autocorrelation Function.

CSV Comma-separated values.

DS Downsampling.

FFT Fast Fourier Transform.

GT Grundton.

GTS Ganztonschritt.

HPS Harmonic Product Spectrum.

HSA Harmonic Spacing Analysis.

HTS Halbtonschritt.

IDE Integrated development environment.

JSON JavaScript Object Notation.

MIDI Musical Instrument Digital Interface.

OT Oberton.

SDF Square Difference Function.

TP Tiefpassfilter.

WAV Waveform Audio File Format.

ZCR Zero-Crossing Rate.

1 Einleitung

1.1 Kontext

Da die meisten Orgeln in Kirchenräumen aufgestellt sind, sind sie dort im Verlauf des Jahres deutlichen Temperaturschwankungen ausgesetzt. Einerseits bedingt durch den jahreszeitliches Temperaturverlauf, als auch durch regelmäßig stattfindende Aufheiz- und Abkühlungsvorgänge, hauptsächlich im Winter. Diese Schwankungen haben direkten Einfluss auf die Stimmhöhe des Instruments.

Dabei zeigt sich das Problem, dass sich die verschiedenen Pfeifenarten unterschiedlich verstimmen. Bei den Labialpfeifen ändert sich die Tonhöhe gleichmäßig, so dass diese auch weiterhin harmonisch miteinander erklingen können, wenn sich die Umgebungstemperatur ändert. Bei den Lingualpfeifen hingegen ist dies nicht den Fall. Deren Tonhöhe bleibt annähernd konstant, weshalb sie im Zusammenspiel mit den Labialpfeifen dann verstimmt klingen. Um das zu vermeiden, werden letztere regelmäßig nachgestimmt.

Dieses Nachstimmen erfolgt nach Gehör. Dazu werden je eine Labial- und Lingualpfeife des gleichen Tons (in derselben oder auch unterschiedlichen Tonlagen) gleichzeitig zum Erklingen gebracht. Die Labialpfeife dient dabei als Referenzton. Wenn beide Töne nicht harmonisch zueinander sind, entsteht eine akustische Schwebung. Die Tonhöhe der Lingualpfeife wird nun so lange verändert, bis keine Schwebung mehr zu hören ist und folglich beide Pfeifen in Resonanz klingen.

1.2 Ziele der Arbeit und Hypothesen

Für diesen speziellen Anwendungsfall existieren bisher keine Hilfsmittel, die den Orgelstimmer bei der Erkennung der Frequenzabweichung und damit beim Stimmvorgang unterstützen könnten. Ziel dieser Arbeit soll es deshalb sein, einen Algorithmus zu entwickeln, um akustische Schwebungen von gleichzeitig erklingenden Labial- und Lingualpfeifen zu erkennen. Dieser soll für übergebene Audiodaten die darin vorhandene Schwebungsfrequenz ermitteln und zurückgeben.

Dieser Anwendungsfall eignet sich für eine Untersuchung besonders gut, weil bei Orgeln Dauertöne mit praktisch unbegrenzter Länge zur Verfügung stehen. Somit kann, im

Gegensatz zu z.B. Klavier- oder Gitarrentönen, die Hüllkurve der Audiosignals vorerst vernachlässigt werden.

Im Rahmen dieser Arbeit sollen die folgenden Hypothesen überprüft werden:

- **H1:** Es kann ein Algorithmus entwickelt werden, mit dem akustische Schwebungen, wie sie bei gleichzeitig erklingenden Orgelpfeifen auftreten, mit einer Trefferquote und Genauigkeit von mehr als 60 % erkannt werden.
- **H2:** Durch den Einsatz von Frequenzerkennungsverfahren, die im Frequenzbereich arbeiten, also das Spektrum von Audiodaten analysieren, können bessere Ergebnisse als bei der Verwendung von Zeitbereichsverfahren erzielt werden.
- **H3:** Der entwickelte Algorithmus kann für die Entwicklung eines Stimmgeräts verwendet werden, das zu jedem Zeitpunkt die aktuelle Audiofrequenz, Schwebungsfrequenz und die notwendige Stimmrichtung, also ob die Pfeife höher oder tiefer gestimmt werden muss, anzeigt.

1.3 Gliederung der Arbeit

Die vorliegende Arbeit ist folgendermaßen gegliedert. Im zweiten Kapitel findet sich ein Überblick über den technischen Hintergrund. Dies schließt u.a. die mathematischen Grundlagen sowie den Aufbau und die Funktionsweise von Orgeln ein. Danach findet sich ein Überblick über den Stand der Technik mit Fokus auf existierende Stimmgeräte und verschiedene Algorithmen zur Frequenzerkennung. Im vierten Kapitel werden der vorgeschlagene Algorithmus zur Schwebungserkennung und Details zur umgesetzten Implementierung vorgestellt. Danach findet sich ein Kapitel mit dem Überblick über die verwendeten Testdaten sowie deren Generierung. Diese werden im sechsten Kapitel zur Evaluation des entwickelten Algorithmus verwendet. Abschließend wird ein Fazit gezogen.

2 Technischer Hintergrund

Dieses Kapitel soll einen Überblick über den technischen Hintergrund geben. Dazu werden für die Arbeit relevante musiktheoretische Grundlagen, Grundlagen des Orgelbaus sowie physikalische Grundlagen genauer erläutert.

2.1 Musiktheoretische Grundlagen

2.1.1 Aufbau der Klaviatur

Bei der Orgel handelt es sich um ein Tasteninstrument. Dementsprechend erfolgt das Spielen von Tönen, wie z.B. auch bei Klavieren und Keyboards, durch das Drücken der Tasten der Klaviatur. Jede Taste entspricht dabei einem anderen Ton. Der schematische Aufbau einer Klaviatur ist in Abbildung 2.1 dargestellt.

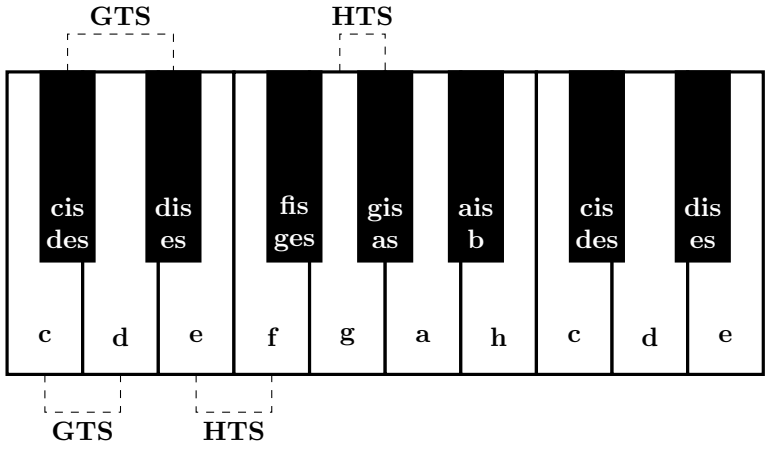


Abbildung 2.1: Schematischer Aufbau einer Klaviatur (nach [Joa15], S. 26).

Sie besteht aus sieben verschiedenen Stammtönen (c, d, e, f, g, a und h), die sich periodisch wiederholen. Diese werden durch die weißen Tasten repräsentiert. Dazwischen befinden sich fünf, von diesen abgeleitete, Töne (cis/des, dis/es, fis/ges, gis/as und ais/b), repräsentiert durch die schwarzen Tasten. Die Abstände zwischen den einzelnen Tönen

werden in Halb- und Ganztonschritte unterschieden. Ein Halbtonschritt (HTS) bezeichnet den Abstand von einer Taste zur direkt benachbarten. Beispiele hierfür sind die Schritte e – f oder g – gis. Als Ganztonschritt (GTS) wir hingegen der Abstand von einer Taste zur übernächsten bezeichnet, wie z.B. die Schritte c – d oder cis – dis.

2.1.2 Diatonische Tonsysteme und Intervalle

In der westlichen Musik kommen üblicherweise diatonische Tonsysteme zum Einsatz, also Systeme bei denen sich die verwendeten Tonleitern aus 5 Ganz- und 2 Halbtonschritten zusammensetzen. Der bekannteste Vertreter hierfür ist das heute vorherrschende Dur-Moll-System. Bei Dur-Tonleitern befinden sich die Halbtöne zwischen dem dritten und vierten sowie dem siebten und achtem Ton. Bei Moll-Tonleitern hingegen zwischen dem zweiten und dritten sowie dem fünften und sechsten Ton. Spielt man bei einer Klaviatur nur die weißen Tasten, so entspricht dies der Tonart C-Dur (Dur-Tonleiter mit c als Grundton) bzw. a-Moll (Moll-Tonleiter mit a als Grundton). Das Spielen anderer Tonarten ist unter Einbeziehung der schwarzen Tasten möglich.

Der Abstand zwischen zwei beliebigen Tönen kann über die sogenannten Intervalle bestimmt werden. Diese geben eine Auskunft, wie weit diese Töne relativ zueinander entfernt sind. Die wichtigsten Intervalle sowie die dazugehörigen Abstände in Halbtönen sind in Tabelle 2.1 gegeben.

Intervall	**Typ**	**Abstand**
Prime	rein	0 HTS
Sekunde	klein	1 HTS
	groß	2 HTS
Terz	klein	3 HTS
	groß	4 HTS
Quarte	rein	5 HTS
Quinte	rein	7 HTS
Sexte	klein	8 HTS
	groß	9 HTS
Septime	klein	10 HTS
	groß	11 HTS
Oktave	rein	12 HTS

Tabelle 2.1: Überblick über Intervalle (nach [Joa15], S. 29).

Die Intervalle können in zwei Gruppen eingeteilt werden: reine und nichtreine Intervalle. Reine Intervalle haben immer denselben festgelegten Abstand. Zu ihnen zählen Prime, Quarte, Quinte und Oktave. Nichtreine Intervalle treten als große und kleine Intervalle auf. Der Unterschied zwischen beiden beträgt dabei jeweils einen Halbton. Zu den nichtreinen Intervallen zählen Sekunde, Terz, Sexte und Septime. Außerdem können alle Intervalle auch vermindert oder übermäßig auftreten, wobei jeweils ein Halbton abgezogen bzw. hinzugefügt wird.

2.1.3 Stimmungssysteme

Obertonreihe und natürliche Stimmung

Die Zuordnung der einzelnen Töne zu den realen Frequenzen, die diesen entsprechen, wird als Stimmung bezeichnet. Bei der natürlichen Stimmung beruht diese Zuordnung auf festgelegten Frequenzverhältnissen, die den einzelnen Intervallen zugeordnet sind. Diese Verhältnisse ergeben sich aus der Obertonreihe. So entspricht der erste Oberton der Oktave (Verhältnis 2 : 1), der zweite Oberton der Quinte (Verhältnis 3 : 2), der dritte Oberton der Quarte (Verhältnis 4 : 3) usw. Ein Überblick über alle Frequenzverhältnisse ist in Tabelle 2.2 gegeben.

Intervall	**Verhältnis**
Prime	1 : 1
kl. Sekunde	16 : 15
gr. Sekunde	9 : 8
kl. Terz	6 : 5
gr. Terz	5 : 4
Quarte	4 : 3

Intervall	**Verhältnis**
Quinte	3 : 2
kl. Sexte	8 : 5
gr. Sexte	5 : 3
kl. Septime	7 : 4
gr. Septime	15 : 8
Oktave	2 : 1

Tabelle 2.2: Frequenzverhältnisse in der natürlichen Stimmung (nach [Mar]).

Diese natürliche Stimmung ist für den Einsatz bei Tasteninstrumenten nicht optimal (vgl. [Wol91], S. 27–28.). Ein Grund dafür ist, dass die Verhältnisse zwischen benachbarten Halbtönen nicht immer gleich groß sind. Das führt dazu, dass das Instrument nur in der Tonart, in der es gestimmt wurde (abhängig vom verwendeten Grundton), rein klingt. In allen anderen Tonarten kommt es zur Inharmonizität.

Ein weiteres Problem besteht darin, dass die Frequenzverhältnisse der natürlichen Stimmung rechnerisch nicht exakt aufgehen. Rein theoretisch müssten 12 Quinten exakt 7 Oktaven entsprechen, da

$$12 \cdot 7\,\mathrm{HTS} = 7 \cdot 12\,\mathrm{HTS} \tag{2.1}$$

gilt. Dies ist jedoch nicht der Fall. Für 7 Oktaven ergibt sich exakt ein Frequenzverhältnis von 128 : 1:

$$\left(\frac{2}{1}\right)^7 = 128. \tag{2.2}$$

Für 12 Quinten hingegen ist dieses größer und damit auch die resultierende Frequenz höher:

$$\left(\frac{3}{2}\right)^{12} \approx 129{,}75. \tag{2.3}$$

Diese Abweichung entspricht ungefähr einem Verhältnis von 74 : 73 und wir als pythagoreisches Komma bezeichnet. Für die Verwendung bei Tasteninstrumenten muss deshalb ein alternatives Stimmungssystem verwendet werden.

Gleichstufige Stimmung

Heute findet hauptsächlich die gleichstufige Stimmung Anwendung. Dabei werden die Intervalle so angepasst, dass das Frequenzverhältnis benachbarter Halbtöne immer gleich groß ist. Lediglich die Oktave (mit ihrem Verhältnis 2 : 1) wird weiterhin rein gestimmt. Alle anderen Intervalle werden dazwischen gleichmäßig verteilt. Für das Frequenzverhältnis v_{HTS} eines einzelnen Halbtons ergibt sich somit folgender Wert:

$$v_{\mathrm{HTS}} = \sqrt[12]{2} \approx 1{,}0595. \tag{2.4}$$

Die entsprechenden Frequenzverhältnisse höherwertige Intervalle können durch Potenzieren dieses Wertes berechnet werden. Das Frequenzverhältnis v zweier Töne mit dem Abstand d_{HTS} (in Halbtönen) berechnet sich folgendermaßen:

$$v = \left(\sqrt[12]{2}\right)^{d_{\mathrm{HTS}}} = 2^{\frac{d_{\mathrm{HTS}}}{12}}. \tag{2.5}$$

Ein Vergleich der Frequenzverhältnisse zwischen der gleichstufigen und natürlichen Stimmung ist in Abbildung 2.2 gegeben.

Um verschiedene Stimmungssysteme vergleichen zu können, wurde die Maßeinheit Cent eingeführt (vgl. [Wol91], S. 29). Diese unterteilt jede Oktave, die ja in allen Systemen rein gestimmt wird, in 1.200 Cent. Bei der gleichstufigen Stimmung entspricht somit jeder der 12 Halbtöne exakt 100 Cent. Basierend auf den oben vorgestellten Formeln lässt sich folgender Zusammenhang für das Frequenzverhältnis v zweier Töne mit dem Abstand d_{c} (in Cent) ableiten:

$$v = \left(\sqrt[12]{2}\right)^{\frac{d_{\mathrm{c}}}{100}} = 2^{\frac{d_{\mathrm{c}}}{12 \cdot 100}}. \tag{2.6}$$

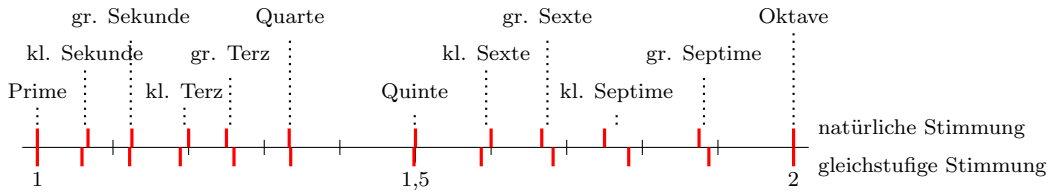


Abbildung 2.2: Vergleich der Frequenzverhältnisse von natürlicher und gleichstufiger Stimmung. Die roten Markierungen entsprechen den Frequenzverhältnissen der verschiedenen Intervalle.

2.2 Orgeln

2.2.1 Aufbau

Die Tonerzeugung bei Orgeln erfolgt dadurch, dass Pfeifen mit Luft durchströmt werden und somit einen Ton erzeugen. Es existieren verschiedene Arten von Orgelpfeifen, die sich u.a. durch ihren grundlegenden Aufbau (Labialpfeifen, Lingualpfeifen, offen, gedackt), das verwendete Material (Metall, Holz) oder ihre Tonlage unterscheiden. Von jeder der verbauten Pfeifenarten existiert jeweils exakt eine Pfeife für jede Taste der Klaviatur. Diese Verbünde von Pfeifen gleichen Typs werden allgemein als Orgelregister bezeichnet.

Die Angabe der Tonhöhe der Register erfolgt als sogenannte Fußtonzahl (vgl. [Wol91], S. 56). Diese Zahl gibt bei offenen Labialpfeifen die ungefähre Länge der tiefsten Pfeife des Registers in Fuß an. Die 8'-Lage entspricht dabei der normalen Stimmhöhe, wo der Kammerton a^1 eine Frequenz von ca. 440 Hz hat. Kleinere Fußtonzahlen entsprechen höher klingenden Registern und größere Fußtonzahlen tiefer klingenden Registern.

Jedes Register kann über Registerwippen oder Zugriegel am Spieltisch einzeln zu- oder abgeschaltet werden. Somit kann der Organist steuern, welche Register erklingen sollen und welche nicht.

Am Spieltisch befinden sich außerdem noch die Klaviaturen, wovon in den allermeisten Fällen mehrere verbaut sind. Das ermöglicht es, verschiedene Klangfarben gleichzeitig einzusetzen und somit z.B. Solostimmen besonders hervorzuheben. Die mit den Händen gespielten Klaviaturen werden als Manuale bezeichnet. Im Gegensatz dazu existiert noch das sogenannte Pedal, eine zusätzliche Klaviatur, die mit den Füßen gespielt wird. Einsatz findet sie üblicherweise als Bassstimme.

Jedem Manual sowie dem Pedal ist eine feste Menge der verbauten Register zugeordnet, die mit diesen verbunden von diesen aus zum Erklingen gebracht werden können. Diese Zusammenfassung von Klaviatur und Registern wird als Werk bezeichnet. Übliche Be-

(a) Überblick.

(b) Pedal.

(c) Registerwippen.

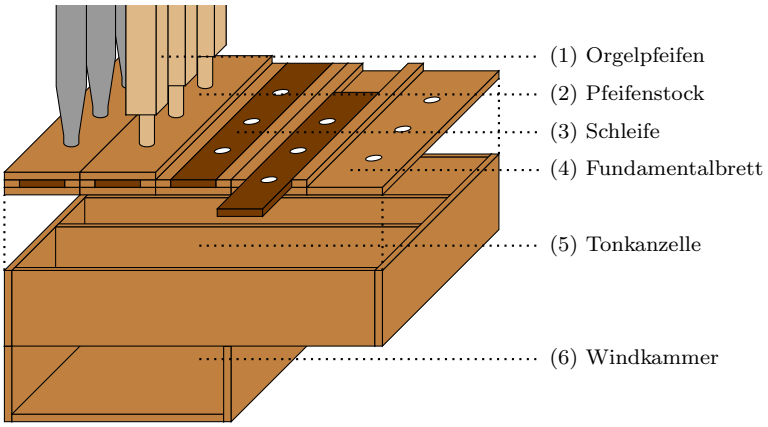
Abbildung 2.3: Spieltisch einer Orgel.

zeichnung für die einzelnen Werke sind Hauptwerk (für das untere Manual), Schwellwerk (für darüberliegende Manuale) und Pedalwerk.

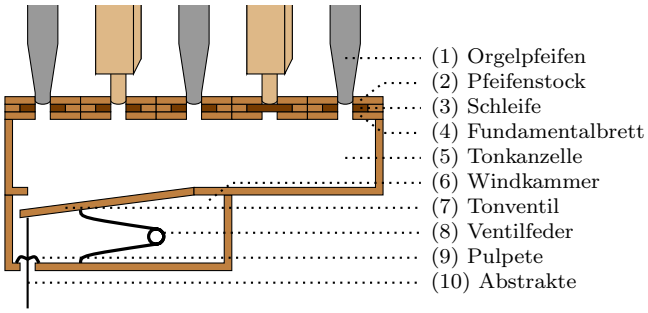
Zusätzlich sind häufig verschiedene Spielhilfen verbaut. Dazu zählen unter anderem Koppeln oder Registrierhilfen. Erstere ermöglichen es zwei Werke miteinander zu verbinden, so dass zusätzlich Pfeifen aus anderen Werken ebenfalls zum Erklingen gebracht werden können. Dadurch kann z.B. ein vollerer Klang erreicht werden. Mithilfe der Registrierhilfen ist es möglich Registrierungen, also den Aktivierungszustand der Orgelregister, zuvor festzulegen und später schnell umzuschalten.

2.2.2 Funktionsweise

Die Tonerzeugung in der Orgel basiert auf einem doppelten Sperrsystem. Eine Pfeife wird nur dann von Luft durchströmt, wenn die entsprechende Taste auf der Klaviatur gedrückt und das dazugehörige Register aktiviert ist. Zur mechanischen Umsetzung dieses Aufbaus kommen verschiedene Bauformen zum Einsatz. Am häufigsten verwendet wird die sogenannte Schleiflade, deren Funktionsweise im Folgenden erläutert wird (vgl. [Wol91], S. 129–132). Eine schematische Darstellung ist in Abbildung 2.4 zu finden.



(a) Aufbau einer Schleiflade.



(b) Querschnitt durch eine Tonkanzelle.

Abbildung 2.4: Aufbau einer Schleifladenorgel (nach [Wol91], S. 131).

.

Die Orgelpfeifen sind in einer matrixähnlichen Form angeordnet. Alle Pfeifen eines Registers (1) sind nebeneinander angeordnet. Mehrere Register werden so hintereinander platziert, dass die gleichen Töne jeweils auf einer gemeinsamen Reihe stehen.

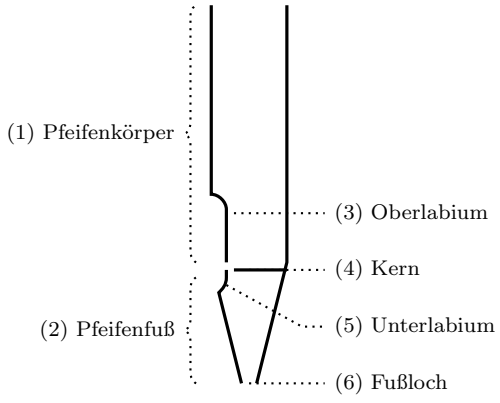
Alle Pfeifen der gleichen Töne sind über sogenannte Tonkanzellen (5) miteinander verbunden. Wenn eine dieser Kanzellen mit Luft durchströmt wird, so erklingen alle darin

enthaltenen Pfeifen. Zur Steuerung dessen dienen die Tonventile (7, 8). Diese trennen die ständig von Luft durchströmte Windkammer (6) von den einzelnen Tonkanzellen. Über die Abstrakten (10) sind sie direkt mit den entsprechenden Tasten der Klaviatur verbunden. Wird eine dieser gedrückt, so öffnet sich das Ventil, die Tonkanzelle wird mit Luft versorgt und die Pfeifen können erklingen.

Der Pfeifenstock (2) dient mit seinen Bohrungen als Halterung für die Orgelpfeifen. Zur Steuerung der erklingenden Register finden die ebenfalls mit Bohrungen versehenen Fundamentalbretter (4) und Schleifen (3) Anwendung. Die in der Mitte angeordnete Schleife kann dabei verschoben werden. Deren Position wird über die an der Orgel verbauten Zugriegel oder Registerwippen gesteuert. Liegen alle Bohrungen übereinander, so ist das Register aktiviert, die Luft kann hindurch strömen und die Pfeife wird zum Erklingen gebracht. Im entgegengesetzten Fall ist das Register deaktiviert und die Luftströmung wird blockiert. Somit bleibt auch die Pfeife stumm.

2.2.3 Labialpfeifen

Bei Labialpfeifen, auch Lippenpfeifen genannt, erfolgt die Tonerzeugung durch eine schwingende Luftsäule (vgl. [Wol91], S. 43–44). Der Aufbau eine solchen Pfeife ist in Abbildung 2.5 dargestellt.



(a) Schematischer Aufbau (nach [Wol91], S. 44).

(b) Foto.

Abbildung 2.5: Aufbau einer Labialpfeife.

Dieser wird in Pfeifenfuß (2) und Pfeifenkörper (1) unterteilt. Während im Pfeifenkörper die Luft schwingt und somit den resultierenden Ton erzeugt, dient der Pfeifenfuß der Windführung. Durch das Fußloch (6) strömt die Luft in diesen ein und wird von

dort durch den Spalt zwischen Kern (4) und Unterlabium (5) gepresst. Der resultierende Luftstrom trifft auf das Oberlabium (3) und regt dadurch die Luft im Pfeifenkörper zur Resonanz an. Durch die Reflexion der Luftbewegung am oberen und unteren Ende des Pfeifenkörpers bildet sich eine stehende Welle in diesem, die nach außen als Ton wahrnehmbar ist.

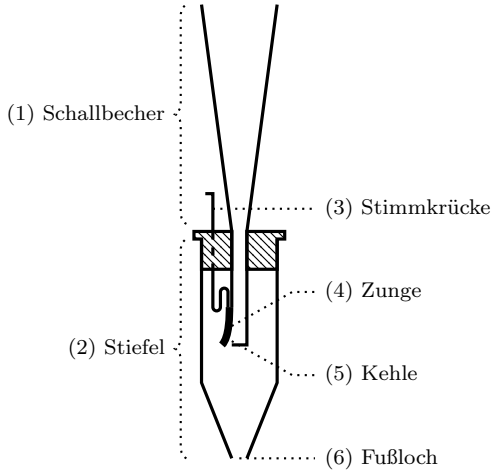
Die Frequenz des resultierenden Tons ist abhängig von der Zeit, die der Luftstrom benötigt, um sich durch den gesamten Pfeifenkörper zu bewegen. Je länger die Luft benötigt, desto länger wird die Periodendauer und desto tiefer der Ton. Diese Zeit hängt hauptsächlich von zwei Faktoren ab: der Höhe des Pfeifenkörpers und der Geschwindigkeit der Luft. Die Höhe der Pfeifen wird durch den Orgelbauer beim Bau festgelegt, so dass diese die gewünschten Frequenzverhältnisse zueinander besitzen, und kann im Nachhinein nicht mehr einfach verändert werden. Die Ausbreitungsgeschwindigkeit der Luft hingegen hängt direkt von der Umgebungstemperatur ab. Je kälter es ist, desto langsamer breitet sich die Luft aus und desto länger benötigt sie für eine festgelegte Strecke. Folglich klingen die Labialpfeifen bei sinkenden Temperaturen tiefer als normal. Im Gegensatz dazu führen höhere Temperaturen auch zu höheren Frequenzen.

2.2.4 Lingualpfeifen

Bei Lingualpfeifen, auch Zungenpfeifen genannt, erfolgt die Tonerzeugung durch eine schwingende Metallzunge (vgl. [Wol91], S. 109–110). Der Aufbau einer solchen Pfeife ist in Abbildung 2.6 dargestellt.

Dieser wird in Stiefel (2) und Schallbecher (1) unterteilt. Im Gegensatz zu den Labialpfeifen dient hier der Stiefel zur Tonerzeugung und der Schallbecher zur Verstärkung dessen. Die Luft strömt aus dem Fußloch (6) durch die Kehle (5) in den Schallbecher. Durch die dabei entstehende Luftbewegung wird die Zunge (4) an die Kehle gezogen und verschließt diese. Somit kann sich keine Luft mehr durch die Pfeife bewegen. Das führt dazu, dass sich die Zunge entspannt und die Kehle wiederum geöffnet wird. Dieser Ablauf wiederholt sich immer wieder und führt damit zu einem periodischen Luftstrom durch die Orgelpfeife, der als Ton wahrnehmbar ist.

Bei Lingualpfeifen hat die Länge des Schallbechers keinen Einfluss auf die Frequenz des resultierenden Tons. Diese hängt lediglich von der freischwingenden Länge der Metallzunge ab. Je länger diese ist, desto tiefer erklingt der Ton. Zur Einstellung der Länge der Metallzunge wird die Stimmkrücke (3) verwendet. Wird diese heruntergeschlagen, so verkürzt sich die Metallzungen und der Ton wird höher. Das Heraufschlagen hingegen führt zu einem tieferen Ton. Im Gegensatz zu den Labialpfeifen hat hier die Umgebungstemperatur keinen Einfluss auf die Tonhöhe.



(a) Schematischer Aufbau (nach [Wol91], S. 110).

(b) Foto.

Abbildung 2.6: Aufbau einer Lingualpfeife.

2.2.5 Vorgehen beim Stimmen

Wie oben dargestellt, führen Temperaturänderungen zu Frequenzänderungen bei den Orgelpfeifen. Während die Stimmhöhe der Lingualpfeifen annähernd gleich bleibt, ändern sich die Frequenzen der Labialpfeifen. Das führt dazu, dass es verstimmt klingt, wenn beide Pfeifenarten gleichzeitig gespielt werden. Um das zu verhindern, ist ein regelmäßiges Nachstimmen notwendig, vor allem beim Übergang vom Sommer in den Winter sowie andersherum. Dabei werden die Lingualpfeifen angepasst, da diese den geringeren Anteil an der Gesamtzahl besitzen, und sich durch die verbaute Stimmkrücke einfacher nachstimmen lassen.

Dafür werden jeweils eine Labial- und Lingualpfeife, die in Resonanz klingen sollten, gleichzeitig angespielt. Ist dies nicht der Fall, so führt die Frequenzabweichung zu einer akustischen Schwebung, also einem periodischen An- und Abschwellen der Lautstärke. Die Stimmhöhe der Lingualpfeife wird nun so lange angepasst, bis keine Schwebung mehr zu hören ist und beide Pfeifen in Resonanz klingen. Dieses Vorgehen wird für alle Lingualpfeifen der Orgel wiederholt. Als Labialregister wird üblicherweise die Oktave 4' verwendet, da diese als sehr stimmstabil gilt (vgl. [Wol91], S. 242).

Zum Stimmen der Lingualpfeifen werden sogenannte Stimmeisen (siehe Abbildung 2.7) verwendet. Mit diesen können die Stimmkrücken herauf- und heruntergeschlagen werden.

Abbildung 2.7: Stimmeisen ([Wei]).

2.3 Schwebungen

Schwebungen entstehen dann, wenn sich zwei Schwingungen (wie z.B. Töne) mit minimal unterschiedlichen Frequenzen f_1 und f_2 überlagern (vgl. [JPM14]). Beispielhaft ist dies in Abbildung 2.8 dargestellt. Durch die minimal unterschiedlichen Frequenzen sind die Maxima und Minima der Schwingungen nicht immer synchron zueinander. Stattdessen verschieben sie sich kontinuierlich, so dass die Signale abwechselnd gleichphasig (B, D) und gegenphasig (A, C, E) zueinander sind (vgl. [Wol91], S. 38). Das führt zur gegenseitigen Verstärkung (B, D) bzw. Auslöschung (A, C, E) der Schwingungen. So entsteht eine sinusförmige Hüllkurve um das Signal.

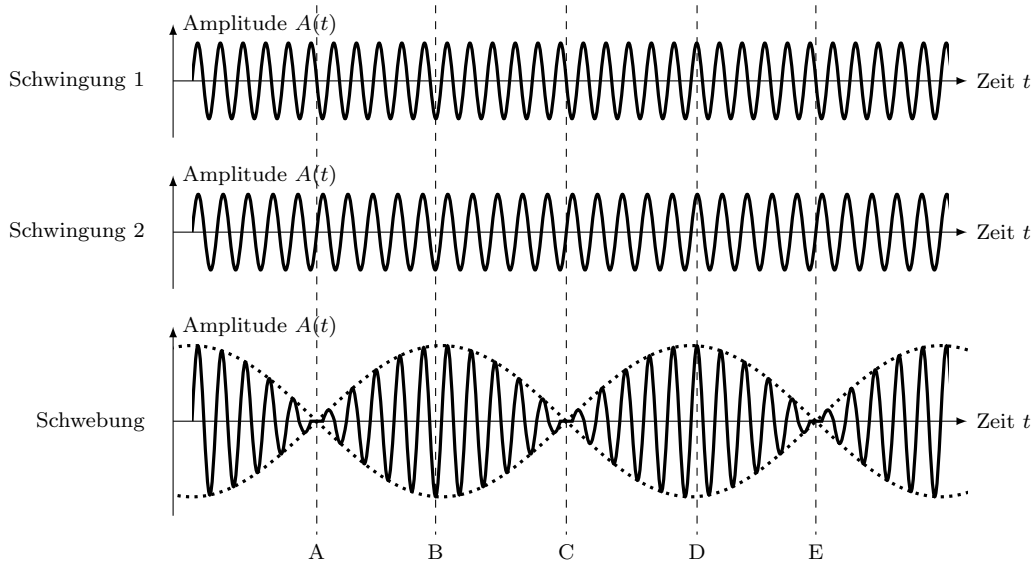


Abbildung 2.8: Entstehung von Schwebungen (nach [Wol91], S. 39).

Die mathematische Grundlagenbetrachtung zu Schwebungen ist in [Jü07], S. 120 f. gegeben. Mathematisch kann eine Schwingung $y(t)$ mit Amplitude $\hat{y}$ und Frequenz f folgendermaßen formalisiert werden:

$$y(t) = \hat{y} \cdot \sin(2\pi f t). \tag{2.7}$$

Die Überlagerung zweier unabhängiger Schwingungen mit gleicher Amplitude $\hat{y}$ und unterschiedlichen Frequenzen f_1 und f_2 kann als Summe modelliert werden:

$$\begin{aligned} y(t) &= y_1(t) + y_2(t) \\ &= \hat{y} \cdot \sin(2\pi f_1 t) + \hat{y} \cdot \sin(2\pi f_2 t) \\ &= \hat{y} \cdot \left(\sin\left(2\pi f_1 t\right) + \sin\left(2\pi f_2 t\right)\right). \end{aligned} \tag{2.8}$$

Mithilfe der Additionstheoreme kann diese Summe folgendermaßen reduziert werden[1]:

$$\begin{aligned} y(t) &= \hat{y} \cdot \left(2 \cdot \sin\left(\frac{2\pi f_1 t + 2\pi f_2 t}{2}\right) \cdot \cos\left(\frac{2\pi f_1 t - 2\pi f_2 t}{2}\right)\right) \\ &= 2\hat{y} \cdot \sin\left(2\pi \frac{f_1 + f_2}{2} t\right) \cdot \cos\left(2\pi \frac{f_1 - f_2}{2} t\right). \end{aligned} \tag{2.9}$$

Das resultierende Signal kann als Produkt aus einem hochfrequenten und einem niederfrequenten Anteil dargestellt werden. Der hochfrequente Sinus-Anteil entspricht der resultierenden Frequenz f_{res}. Diese liegt exakt in der Mitte der beiden Ausgangsfrequenzen f_1 und f_2:

$$f_{\text{res}} = \frac{f_1 + f_2}{2}. \tag{2.10}$$

Der niederfrequente Kosinus-Anteil hingegen kontrolliert die Hüllkurve und damit das periodische An- und Abschwellen der Intensität. Der Abstand zwischen zwei benachbarten Intensitätsminima bzw. -maxima wird mithilfe der Schwebungsfrequenz f_{Schweb} beschrieben. Diese gibt an, wie viele Minima bzw. Maxima pro Sekunde auftreten. Sie ist direkt abhängig vom Abstand der Ausgangsfrequenzen. Je weiter diese auseinander liegen, desto größer ist die Schwebungsfrequenz. Da jede Periode der Kosinus-Funktion zwei Nulldurchläufe besitzt, kommt es während jeder vollständigen Periode der Einhüllenden exakt zweimal zur Auslöschung der Ausgangsschwingungen. Folglich ist die Schwebungsfrequenz doppelt so groß ist wie die Frequenz der Einhüllenden. Für die Schwebungsfrequenz f_{Schweb} gilt somit:

$$f_{\text{Schweb}} = |f_1 - f_2|. \tag{2.11}$$

[1] $\sin(\alpha) + \sin(\beta) = 2 \cdot \sin\left(\frac{\alpha+\beta}{2}\right) \cdot \cos\left(\frac{\alpha-\beta}{2}\right)$ ([FDD+03], S. 26)

Die Betragsfunktion hier rührt daher, dass es sich bei der Kosinusfunktion um eine gerade Funktion handelt, also $\cos(x) = \cos(-x)$ gilt. Folglich hat es auf das resultierende Ergebnis keinen Einfluss, welche der Ausgangsfrequenzen größer bzw. kleiner als die andere ist. Für die Schwebungsfrequenz ist lediglich der Abstand zwischen beiden relevant.

Bei dem in dieser Arbeit betrachteten Anwendungsfall des Stimmens von Lingualpfeifen ist jedoch nicht immer gegeben, dass die beiden Töne wirklich nah beieinander liegende Frequenzen besitzen. Dadurch, dass üblicherweise das Oktave 4'-Register als Referenz verwendet wird, kann es vorkommen, dass die zu stimmenden Pfeifen in tieferen Oktaven liegen (z.B. in 8'-, 16'- oder 32'-Lage). Da Oktavintervalle durch ein Frequenzverhältnis von $2:1$ gekennzeichnet sind, ist garantiert dass das Verhältnis beider Frequenzen annähernd einer Zweierpotenz entspricht:

$$f_1 \approx o \cdot f_2 = 2^n \cdot f_2,\ n \in \mathbb{N}. \tag{2.12}$$

Die Variable o wird im Folgenden als Oktavfaktor bezeichnet. Mit diesem Faktor lassen sich die Formeln 2.10 und 2.11 verallgemeinern. Jede Periode der Frequenz f_2 entspricht annähernd o Perioden von f_1. Aufgrund dieser Kompatibilität kann die Frequenz f_1', die dem Ton von f_1 in der tieferen Oktavlage von f_2 entspricht, berechnet werden:

$$f_1' = \frac{f_1}{o}. \tag{2.13}$$

Damit kann die Formel 2.10 für die resultierende Frequenz f_{res} angepasst werden:

$$f_{\text{res}} = \frac{f_1' + f_2}{2} = \frac{\frac{f_1}{o} + f_2}{2}. \tag{2.14}$$

Die Schwebungsfrequenz f_{Schweb} wird durch die tiefste Frequenz bestimmt, bei der zwei Teiltöne von f_1 und f_2 aufeinandertreffen. Dies ist der Fall für den o-ten Teilton von f_2 und den Grundton von f_1. Für die Frequenz f_2' des o-ten Teiltons gilt:

$$f_2' = o \cdot f_2. \tag{2.15}$$

Somit kann die Formel 2.11 folgendermaßen verallgemeinert werden:

$$f_{\text{Schweb}} = |f_1 - f_2'| = |f_1 - o \cdot f_2|. \tag{2.16}$$

3 Stand der Technik

Dieses Kapitel gibt einen Überblick über den aktuellen Stand der Technik. Zuallererst wird auf verschiedene existierende Stimmgeräte eingegangen. Danach werden verschiedene Verfahren zur monophonen und polyphonen Frequenzerkennung genauer betrachtet.

3.1 Existierende Stimmgeräte

Aktuell existieren nur zwei Stimmsoftwares, die es ermöglichen Schwebungen zu erkennen. Diese sind das *Professionelles Stimmgerät für Akkordeons* und *Dirk's Stimmgerät für chromatische Instrumente* von *Dirk's Projects* (vgl. [Dirb], [Dira]). Für die Schwebungserkennung wird dabei eine Frequenzanalyse, basierend auf einer schnellen Fouriertransformation (Fast Fourier Transform, FFT), verwendet. Die verwendete Fenstergröße entspricht dabei einer Audiolänge von 20 Sekunden (vgl. [Dirb]). Durch diese Genauigkeit können die beiden Ursprungstöne als getrennte Maxima im Frequenzspektrum wahrgenommen werden (siehe Abbildung 3.1). Diese Softwares sind vor allem dafür ausgelegt, dass eine festgelegte Schwebungsfrequenz zwischen zwei Tönen erreicht werden soll, wie es z.B. beim Stimmen von Klavieren oder Akkordeons gemacht wird.

Das Problem bei diesem Ansatz ist, dass Schwebungen unter einem Minimalwert von 0,6 Hz nicht gefunden werden können (vgl. [Dirb]). Bei geringeren Schwebungsfrequenzen kann nicht unterschieden werden, ob eine Schwebung vorliegt oder beide Töne in Resonanz klingen. Aufgrund dieser Einschränkung ist die Software nicht für die Verwendung bei Orgelpfeifen geeignet, da dort genau diese Unterscheidung benötigt wird.

Abgesehen davon arbeiten alle sonstigen Stimmgeräte und -softwares nur monophon, können also nur die Grundfrequenz eines Signals erkennen. Dabei kommen verschiedene Ansätze bzw. Algorithmen, u.a. Fourieranalyse ([AEVS19], [USLR20]), Harmonic Product Spectrum ([MA09]) oder Autokorrelation ([RJ08]) zum Einsatz.

3.2 Monophone Frequenzerkennung

In diesem Kapitel werden verschiedene Algorithmen zur Frequenzerkennung im monophonen Fall betrachtet. Dabei wird davon ausgegangen, dass nur exakt eine Grundfrequenz im Signal vorhanden ist und diese erkannt werden soll. Die existierenden Algorithmen

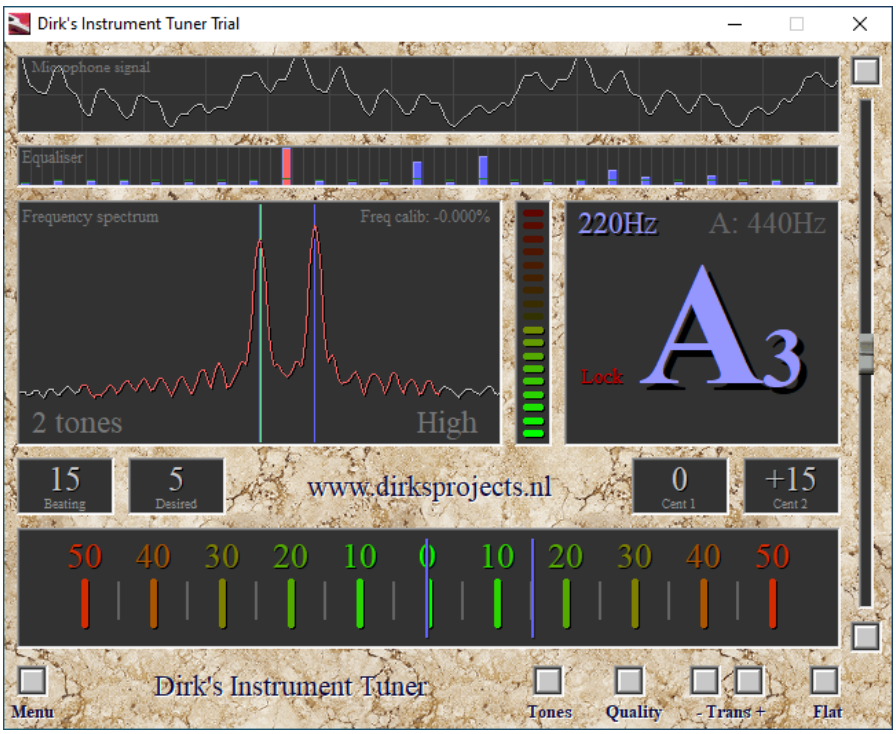


Abbildung 3.1: Benutzeroberfläche von Dirk's Stimmgerät für chromatische Instrumente. In diesem Beispiel wird ein Ausgangssignal mit zwei Ausgangstönen (220 Hz und 222 Hz) verwendet. Diese sind im Frequenzspektrum als getrennte Peaks zu finden. Der Frequenzunterschied von 15 Cent wird sowohl als Zahl ausgegeben sowie grafisch dargestellt.

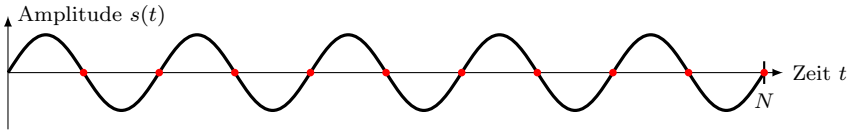
können, entsprechend der verwendeten Ausgangsdaten, in Zeit- und Frequenzbereichsverfahren unterteilt werden. Im Zeitbereich wird ausschließlich der zeitliche Verlauf des Signals analysiert und darüber eine Frequenz ermittelt. Im Frequenzbereich hingegen wird zuvor das Amplitudenspektrum des Ausgangssignals berechnet und dieses analysiert.

Für die Verwendung all dieser Verfahren ist zuvor eine Überführung des analogen Audiosignals in einen zeit- und wertdiskreten Bereich notwendig. Dazu wird das Signal mit einer festgelegten Samplerate f_S entsprechend oft pro Sekunde abgetastet. Die resultierenden Daten können als Eingabe für die Frequenzerkennung verwendet werden.

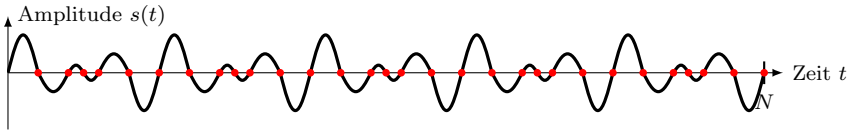
3.2.1 Algorithmen im Zeitbereich

Zero-Crossing Rate

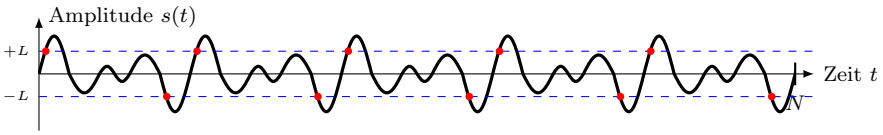
Bei der Verwendung der Zero-Crossing Rate (ZCR) erfolgt die Frequenzerkennung durch Zählen der Nulldurchläufe des Ausgangssignals, wie u.a. in [RJ08] beschrieben. Eine grafische Darstellung ist in Abbildung 3.2 gegeben.



(a) Algorithmus ohne Schwellwert mit reiner Sinuswelle.



(b) Algorithmus ohne Schwellwert mit komplexer Wellenform.



(c) Algorithmus mit Schwellwert mit komplexer Wellenform.

Abbildung 3.2: Vorgehen zur Ermittlung der ZCR mit verschiedenen Ansätzen und Ausgangssignalen.

Dabei wird in einem festgelegten Bereich des Signals $s(t)$ der Länge N die Anzahl der Vorzeichenwechsel N_c gezählt. Darauf basierend kann die Rate der Nulldurchläufe (ZCR) berechnet werden:

$$zcr = \frac{N_c}{N}. \tag{3.1}$$

Durch Multiplikation mit der Samplerate f_S des Signals kann dessen Frequenz f_{ZCR} berechnet werden. Da jede Periode des Signals exakt zwei Nulldurchläufe enthält, ist dabei eine Division durch zwei notwendig:

$$f_{\mathrm{ZCR}} = \frac{zcr \cdot f_S}{2}. \tag{3.2}$$

Für einfache Wellenformen, wie z.B. reine Sinuswellen, funktioniert die ZCR sehr zuverlässig (siehe Abbildung 3.2a). Bei komplexeren Wellenformen kann es vorkommen, dass eine einzelne Periode mehr als zwei Nulldurchläufe enthält (siehe Abbildung 3.2b).

Dies führt zu falschen Ergebnissen für die ZCR als auch für die ermittelte Frequenz. Um diese Probleme zu reduzieren, können zusätzlich Schwellwerte L eingeführt werden, z.B. basierend auf den durchschnittlichen Signalwerten:

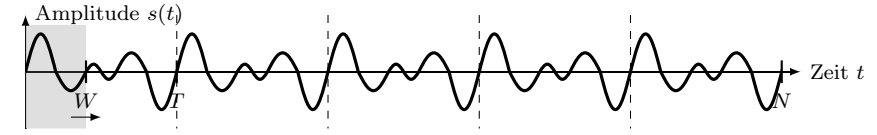
$$L = \pm 1{,}2 \cdot \frac{1}{N} \cdot \sum_{n=0}^{N-1} |s(n)| \qquad ([\text{RJ08}]). \tag{3.3}$$

Damit wird ein Nulldurchlauf nur dann gezählt, wenn dieser Schwellwert über- oder unterschritten wird (siehe Abbildung 3.2c).

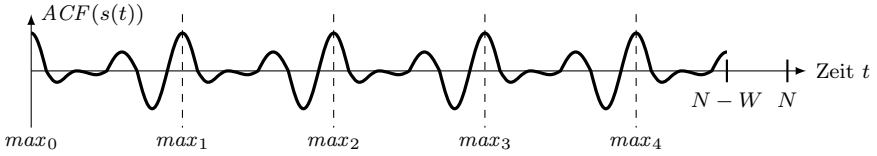
Autokorrelation und quadratische Differenzfunktion

Bei der Autokorrelationsfunktion (Autocorrelation Function, ACF) und der quadratischen Differenzfunktion (Square Difference Function, SDF) handelt es sich um zwei eng miteinander verwandte Funktionen.

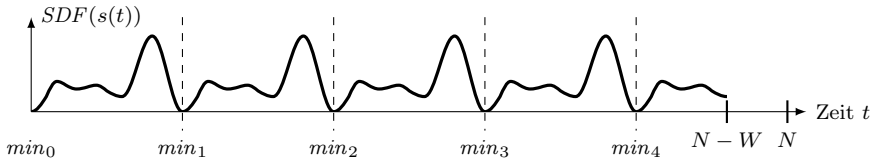
Bei beiden wird die Korrelation des Signals $s(t)$ mit sich selbst berechnet. Dazu wird ein Fenster der Größe W schrittweise über das Signal hinweg verschoben (siehe Abbildung 3.3a). An jeder Position wird die Korrelation zwischen dem aktuellen Fenster und dem allerersten Fenster berechnet. Dabei kommen jedoch unterschiedliche Berechnungsfunktionen zum Einsatz.



(a) Zu analysierendes Audiosignal.



(b) Auf $s(t)$ angewandte Autokorrelationsfunktion.



(c) Auf $s(t)$ angewandte quadratische Differenzfunktion.

Abbildung 3.3: Berechnung von ACF und SDF.

Bei der ACF wird die Summe der Produkte der übereinanderliegenden Werte beider Fenster berechnet (vgl. [RJ08], [MAS03]). Der Wert der ACF zu einem Zeitpunkt t lässt sich folgendermaßen berechnen:

$$ACF(s(t)) = \frac{1}{W} \sum_{i=0}^{W-1} s(i) \cdot s(t+i), \qquad \forall t \in [0, N-W]\,. \tag{3.4}$$

Hierbei führen hohe Übereinstimmungen zwischen den Signalverläufen in den beiden jeweils betrachteten Fenstern zu großen Ergebniswerten (siehe Abbildung 3.3b).

Bei der SDF hingegen wird die Summe der quadratischen Differenzen berechnet (vgl. [PG05]). Für einem Zeitpunkt t gilt folgende Formel:

$$SDF(s(t)) = \frac{1}{W} \sum_{i=0}^{W-1} (s(i) - s(t+i))^2, \qquad \forall t \in [0, N-W]\,. \tag{3.5}$$

Hohe Übereinstimmungen zwischen den Signalverläufen führen hier zu kleinen Ergebniswerten (siehe Abbildung 3.3c).

An den Fensterpositionen mit den höchsten Übereinstimmungen entstehen jeweils Maxima bzw. Minima in den ACF- und SDF-Verläufen. Der Abstand zwischen den einzelnen Extrema $(max_0, max_1, max_2, \ldots)$ bzw. $(min_0, min_1, min_2, \ldots)$ entspricht genau der Periodendauer des Signals $s(t)$:

$$T_{\text{ACF}} = max_1 - max_0 = max_2 - max_1 = max_{i+1} - max_i, \tag{3.6}$$

$$T_{\text{SDF}} = min_1 - min_0 = min_2 - min_1 = min_{i+1} - min_i. \tag{3.7}$$

Die Grundfrequenz f_{ACF} bzw. f_{SDF} des Signals $s(t)$ kann über den Kehrwert berechnet werden:

$$f_{\text{ACF}} = \frac{1}{T_{\text{ACF}}}, \tag{3.8}$$

$$f_{\text{SDF}} = \frac{1}{T_{\text{SDF}}}. \tag{3.9}$$

3.2.2 Algorithmen im Frequenzbereich

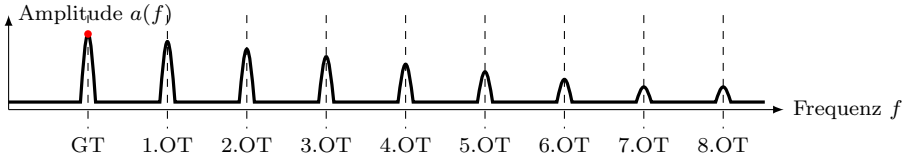
Fourieranalyse

Durch Anwendung der FFT auf das Ausgangssignal $s(t)$ erhält man Informationen über die im Signal vorkommenden Frequenzen und deren Amplituden. Da es sich bei den Signalen nur um reelle Daten handelt, reicht es aus, nur das Amplitudenspektrum $a(f)$ zu betrachten. In diesem wird, wie u.a. in [AEVS19] und [USLR20] beschrieben, nach

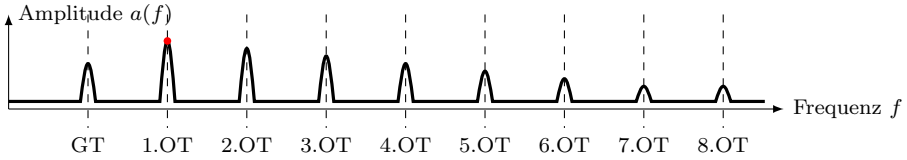
dessen globalem Maximum gesucht. Dieses entspricht im Idealfall der Grundfrequenz f des Signals $s(t)$. Dazu wird der Bin i_{max} mit dem maximalen Wert gesucht. Die dementsprechende Frequenz f_{FFT} kann folgendermaßen berechnet werden:

$$f_{\mathrm{FFT}} = i_{max} \cdot \frac{f_S}{N}. \tag{3.10}$$

f_S und N bezeichnen dabei wieder die Samplerate des Ausgangssignals sowie die Länge des untersuchten Signalausschnitts.



(a) Amplitudenspektrum mit Grundton als globales Maximum.



(b) Amplitudenspektrum mit erstem Oberton als globales Maximum.

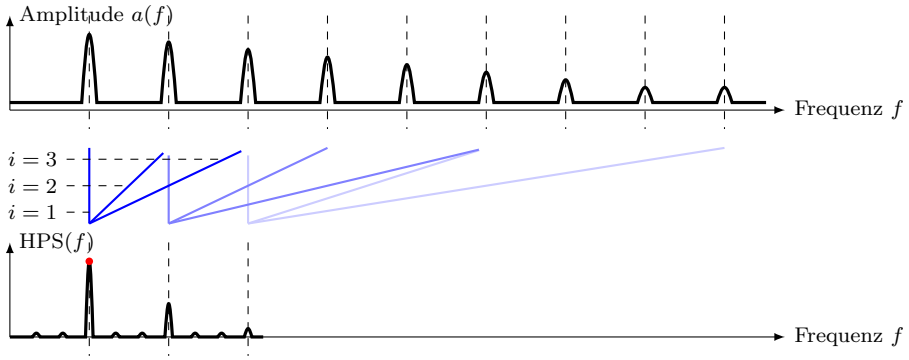
Abbildung 3.4: Frequenzerkennung per FFT (GT: Grundton, OT: Oberton).

Dieses Verfahren funktioniert sehr zuverlässig, wenn der Grundton dem globalen Maximum im Spektrum entspricht (siehe Abbildung 3.4a). Bei komplexeren Wellenformen hingegen kann es vorkommen, dass der Grundton nicht automatisch auch das globale Maximum des Spektrums ist (siehe Abbildung 3.4b). Stattdessen wird ein dazugehöriger Oberton erkannt und die ermittelte Frequenz ist höher als die eigentliche Grundfrequenz. Je nach gefundenem Oberton kann es sich bei den entsprechenden Ergebnissen um dieselben Töne in höheren Oktaven (1. OT, 3. OT, 7. OT, ...) oder komplett andere Töne (2. OT, 4. OT, 5. OT, 6. OT, ...) handeln. Dieses Phänomen tritt unter anderem in verschiedenen Orgelregistern, wie z.B. der Gambe, auf (vgl. [Wol91], S. 59).

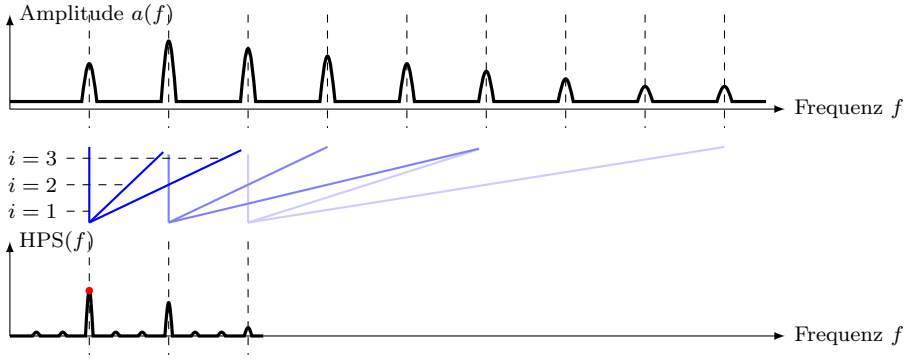
Harmonic Product Spectrum

Die Nachteile der Fourieranalyse können umgangen werden, indem die harmonische Struktur des Amplitudenspektrums $a(f)$ bei der Berechnung berücksichtigt wird (harmonische Töne setzen sich aus Frequenzanteilen des Grundtons und dessen ganzzahligen

Vielfachen zusammen). Dies ist u.a. durch den Einsatz des Harmonic Product Spectrum (HPS) möglich (vgl. [PAC01], [MA09]). Dabei werden jeweils insgesamt H viele Frequenzen, der Grundton und die ersten $H-1$ Obertöne/Vielfachen, betrachtet. Eine Visualisierung ist in Abbildung 3.5 gegeben.



(a) HPS-Berechnung bei einem Amplitudenspektrum mit Grundton als globales Maximum.



(b) HPS-Berechnung bei einem Amplitudenspektrum mit Oberton als globales Maximum.

Abbildung 3.5: Frequenzerkennung per HPS mit $H = 3$.

Für die Berechnung wird bei allen betrachteten Frequenzen bzw. Bins das Produkt dieser H Werte gebildet:

$$HPS(f) = \prod_{i=1}^{H} a(i \cdot f), \qquad \forall f \in \left[0, \frac{f_S}{2H}\right]. \tag{3.11}$$

Für jeden berechneten Wert des HPS müssen immer exakt H Ausgangswerte zur Verfügung stehen. Der H-fache Werte der maximal betrachteten Frequenz muss deshalb noch im zugrundeliegenden FFT-Spektrum enthalten sein. Dementsprechend ist der

Definitionsbereich des HPS gegenüber dem FFT-Spektrum ($\left[0, \frac{f_S}{2}\right]$) um den Faktor H eingeschränkt.

Durch die Produktbildung entstehen im HPS dort die Maxima, wo alleinig lokale FFT-Maxima für die Berechnung verwendet wurden. Dies ist der Fall bei der Grundfrequenz und deren Vielfachen. Da die Intensität der Oberschwingungen im Allgemeinen mit steigender Frequenz abnimmt, führt dies dazu, dass das globale Maximum des HPS in den meisten Fällen bei der Grundfrequenz der Schwingung auftritt.

Die Suche des Ergebnisses erfolgt analog zur FFT. Basierend auf dem Bin i_{max} mit dem globalen Maximum berechnet sich die gefundene Frequenz f_{HPS} folgendermaßen:

$$f_{\text{HPS}} = i_{max} \cdot \frac{f_S}{N}. \tag{3.12}$$

f_S und N bezeichnen hier ebenfalls die Samplerate des zugrundeliegenden Signals sowie die Länge des Signalausschnitts.

Harmonic Spacing Analysis

Die Funktionsweise des HPS basiert darauf, dass alle harmonischen Schwingungen eines einzelnen Tons den gleichen Abstand zueinander haben. Dieser entspricht genau der Grundfrequenz des Signals $s(t)$. Einen direkteren Ansatz nutzt die Harmonic Spacing Analysis (HSA). Sie untersucht das Amplitudenspektrum $a(f)$ direkt nach den lokalen Maxima und analysiert deren Abstände zueinander (vgl. [Ste78]). Eine schematische Darstellung ist in Abbildung 3.6 zu finden.

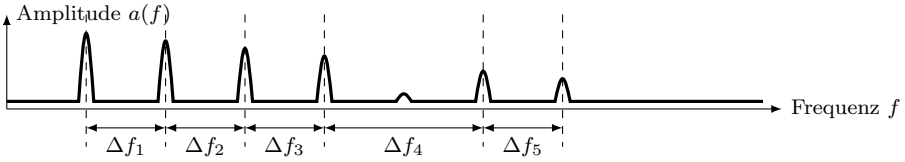


Abbildung 3.6: Frequenzerkennung per HSA. Hier entsprechen die Abstände Δf_1, Δf_2, Δf_3 sowie Δf_5 jeweils der Grundfrequenz des Signals. Der Abstand Δf_4 hingegen ist doppelt so groß, da der 4. OT nicht als Maxima gefunden wurde.

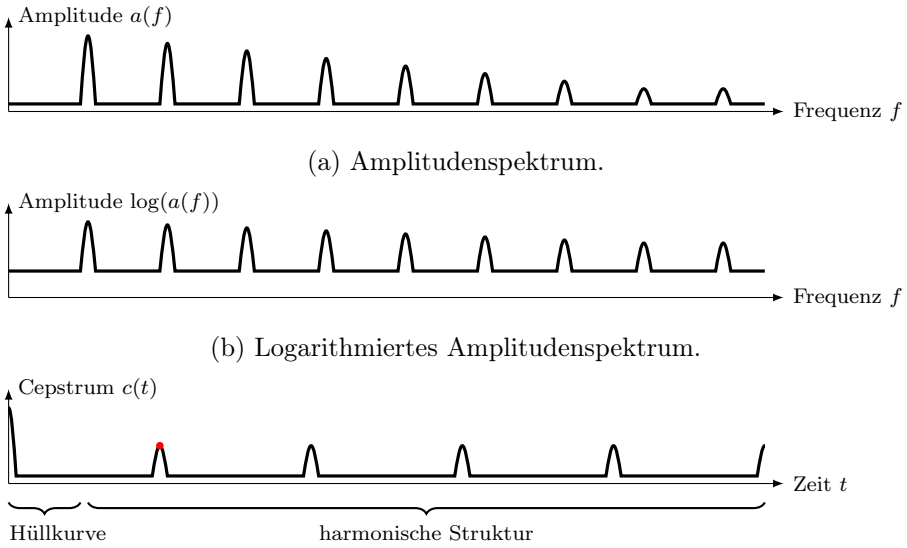
In dem Amplitudenspektrum $a(f)$ werden die lokalen Maxima gesucht und alle Abstände Δf_i zwischen benachbarten Maxima berechnet. Diese entsprechen der gesuchten Grundfrequenz (wenn alle Maxima korrekt erkannt wurden) oder einem ganzzahligen Vielfachen dieser (falls einzelne Maxima nicht gefunden und damit übersprungen wurden). Zum Berechnen des finalen Ergebnisses müssen diese zusammengefasst werden. Dies

kann z.B. durch einen Clustering-Algorithmus geschehen. Dabei werden in einem ersten Schritt alle nah beieinanderliegenden Werte in gemeinsame Klassen einsortiert. Danach werden Klassen, die ganzzahlige Vielfache anderer Klassen sind, zu diesen hinzugefügt. Das finale Ergebnis berechnet sich als der Durchschnittswert der Klasse mit den meisten Elementen.

Der Hauptvorteil der HSA ist, dass, im Gegensatz zum HPS, keine Einschränkung des Suchbereichs stattfindet. Somit können auch Signale, bei denen die Grundfrequenz höher liegt, korrekt erkannt werden.

Cepstrum

Einen ähnlichen Ansatz wie die HSA verfolgt die Frequenzerkennung per Cepstrum. Dort wird die Periodizität der Maxima durch Anwendung einer erneuten Fouriertransformation auf das Amplitudenspektrum $a(f)$ ermittelt (vgl. [PAC01], [LMAM76]). Der Name des Verfahrens ergibt sich durch Umkehren der Buchstaben der ersten Silbe des englischen Wortes *Spectrum*. Durch die zweite Anwendung der FFT kommt es zu einem Übergang vom Frequenzbereich zurück in den Zeitbereich. Der resultierende Definitionsbereich wird analog als Quefrency (abgeleitet vom englischen Wort *Frequency*) bezeichnet. Der Ablauf der Frequenzerkennung per Cepstrum ist schematisch in Abbildung 3.7 dargestellt.



(a) Amplitudenspektrum.

(b) Logarithmiertes Amplitudenspektrum.

(c) Cepstrum des Amplitudenspektrums.

Abbildung 3.7: Frequenzerkennung per Cepstrum.

Für die Berechnung wird das Amplitudenspektrum $a(f)$ zuerst logarithmiert (siehe Abbildung 3.7b). Dadurch wird der absolute Unterschied der verschiedenen Maximalwerte verringert. Von dem resultierenden Spektrum wird erneut das Amplitudenspektrum berechnet. Das resultierende Cepstrum $c(t)$ (siehe Abbildung 3.7c) enthält sowohl Informationen über die Hüllkurve des Amplitudenspektrums sowie dessen harmonische Struktur. Die Hüllkurve, also die Funktion, die den Verlauf der Maxima des Amplitudenspektrums beschreibt und folglich eine entsprechend geringe Frequenz hat, ist im niederen Quefrency-Bereich repräsentiert. Die höherfrequenten Anteile hingegen entsprechen der harmonischen Struktur des Spektrums, also den Abständen zwischen den einzelnen Maxima. Das erste lokale Maximum in diesem Bereich entspricht der Grundfrequenz des Signals.

Die Formel zur Bestimmung der Frequenz aus einem Quefrency-Index kann aus den auch bereits oben verwendeten Formeln für die Fourieranalyse abgeleitet werden:

$$f = i_{\max} \cdot \frac{f_S}{N}. \tag{3.13}$$

Für das Cepstrum kann das folgendermaßen übertragen werden:

- $f \rightarrow T$: Da sich der Wertebereich des Cepstrum im Zeitbereich befindet, erhält man als Ergebnis keine Frequenz, sondern eine Periodendauer.
- $i_{\max}$: Entspricht dem Index des Bins mit maximalem Wert im Cepstrum.
- $f_S \rightarrow \frac{N}{f_S}$: Statt der Samplerate wird hier die Frequenz der FFT-Bins, also die inverse Breite eines einzelnen Bins, verwendet. Diese berechnet sich aus der Samplerate f_S und der Breite des untersuchten Signals N.
- $N \rightarrow N_{\text{Cep}}$: Hier wird die Fenstergröße, die zur Berechnung der zweiten Fouriertransformation verwendet wurde, genutzt.

Damit ergibt sich für die gefundene Periodendauer T_{Cep} an einer Position $i_{\max}$ folgende Formel:

$$T_{\text{Cep}} = i_{\max} \cdot \frac{\frac{N}{f_S}}{N_{\text{Cep}}} = i_{\max} \cdot \frac{N}{N_{\text{Cep}}} \cdot \frac{1}{f_S}. \tag{3.14}$$

Die dazugehörige Frequenz f_{Cep} berechnet sich als deren Reziprokes:

$$f_{\text{Cep}} = \frac{1}{T_{\text{Cep}}} = \frac{1}{i_{\max}} \cdot \frac{N_{\text{Cep}}}{N} \cdot f_S. \tag{3.15}$$

3.3 Polyphone Frequenzerkennung

Neben den monophonen Verfahren zur Frequenzerkennung gibt es verschiedene Ansätze, dies auf den polyphonen Fall zu übertragen. Dabei sollen mehrere gleichzeitig gespielte Töne erkannt werden. Diese Verfahren sind vor allem für die automatische Transkription von Musik interessant, um automatisiert Noten aus einer Audioaufnahme generieren zu können (vgl. [CAX09]). Dabei liegt der Fokus darauf, zu erkennen, zu welchem Zeitpunkt welche Töne gespielt werden.

[TI18] analysiert dafür z.B. die Maxima im FFT-Amplitudenspektrums mit einem iterativen Vorgehen. Jedes Maximum, dessen Wert größer als ein festgelegter Schwellwert ist, wird als gefundener Ton klassifiziert. Danach werden die Frequenzen des Maximums und seiner Obertöne aus dem Spektrum subtrahiert. Somit verbleiben nur die restlichen Töne im Spektrum. Dieses Vorgehen wird so lange wiederholt bis dieses keine Maxima mehr enthält.

[SU15] analysiert das FFT-Amplitudenspektrum durch Suchen dessen Maxima sowie durch die Berechnung mehrerer Autokorrelationen auf diesem. Beides wird unabhängig voneinander berechnet und die Ergebnisse dann verknüpft. Ein Ergebnis ist nur gültig, wenn es durch beide Ansätze gefunden wurde.

Weitere Algorithmen arbeiten ebenfalls auf Basis des Amplitudenspektrums (vgl. [CAX09], [YD07]). All diese Ansätze sind für die Erkennung größerer Frequenzunterschiede (z.B. unterschiedliche Töne auf einer Klaviatur) gedacht und deshalb nicht für die Erkennung von Schwebungen geeignet. Wie bei den existierenden Stimmgeräten (siehe Kapitel 3.1) kann das Spektrum nicht mit ausreichender Genauigkeit berechnet werden. Um bei der Berechnung eine Auflösung von 0,05 Hz zu erreichen, benötigt man 20 Sekun-

den Audiodaten[1]. Dies würde zu einer extremen Verzögerung zwischen Audiosignal und Ergebnis führen, da diese Zeit gewartet werden müsste, um ein valides Ergebnis zu erhalten. Ebenso würden Änderungen der Schwebungsfrequenz innerhalb dieser Zeitspanne nicht registriert werden können.

3.4 Zusammenfassung

Die oben präsentierte Recherche zum Stand der Technik hat gezeigt, dass bisher keine Algorithmen oder Systeme existieren, um die Schwebungsfrequenz von gleichzeitig erklingenden Orgelpfeifen zu erkennen. Existierende Stimmgeräte für ähnliche Anwendungsfälle bieten aufgrund des verwendeten Analyseverfahrens keine ausreichende Genauigkeit (siehe Kapitel 3.1). Alle sonstigen Stimmgeräte verwenden Verfahren der monophonen Frequenzerkennung (siehe Kapitel 3.2). Diese könnten dazu verwendet werden, die resultierende Frequenz der beiden Ausgangstöne zu bestimmen. Sie geben aber keine Informationen über die Schwebungsfrequenz. Für die polyphone Frequenzerkennung existieren verschiedene Verfahren (siehe Kapitel 3.3), vor allem aus dem Bereich der automatischen Musiktranskription. Dort wird aber nach größeren Frequenzabständen, wie sie zwischen verschiedenen Tönen der Tonleiter vorkommen, gesucht. Für den hier betrachteten Anwendungsfall, mit Abständen im Bereich weniger Cent, ist diese Genauigkeit nicht ausreichend.

Deshalb muss für die gewünschte Erkennung der Schwebungen ein neuer Ansatz gewählt werden. Dieser wird im folgenden Kapitel genauer beschrieben.

[1] Der minimal erkennbare Frequenzunterschied Δf eines FFT-Amplitudenspektrums entspricht dem Verhältnis der Samplerate f_S zur verwendeten Datengröße N. Dies folgt direkt aus Formel 3.10 zur Berechnung der Frequenzen eines FFT-Amplitudenspektrums:

$$\begin{aligned} \Delta f &= i \cdot \frac{f_S}{N} - (i-1) \cdot \frac{f_S}{N} \\ &= (i - (i-1)) \cdot \frac{f_S}{N} \\ &= \frac{f_S}{N}. \end{aligned} \tag{3.16}$$

Folglich besteht zwischen Δf und N ein indirekt proportionaler Zusammenhang. Um eine Genauigkeit von 0,05 Hz zu erreichen, gilt für die untere Grenze der Datengröße N folgendes:

$$0{,}05\,\text{Hz} = \frac{f_S}{N} \Leftrightarrow N = \frac{f_S}{0{,}05\,\text{Hz}} = 20\,\text{s} \cdot f_S. \tag{3.17}$$

Es werden also Eingabedaten einer Gesamtlänge von mindestens 20 s für die gewünschte Genauigkeit benötigt.

4 Implementierung

4.1 Überblick

Für das Erkennen von vorliegenden Schwebungen, und damit implizit auch für das Ermitteln der dazugehörigen Frequenzabweichungen, sind zwei Schritte notwendig. Zuerst muss die Frequenz des Audiosignals ermittelt werden. Diese entspricht der resultierenden Frequenz der beiden Ausgangssignale. Außerdem muss die Frequenz der vorliegenden Schwebung bestimmt werden. Beide Informationen können zusammen verwendet werden, um die Frequenzen der beiden Ausgangssignale zu berechnen (siehe Kapitel 2.3). Diese Schritte werden auch in der verwendeten Implementierung so angewendet. Das dazugehörige Datenflussdiagramm ist in Abbildung 4.1 gegeben.

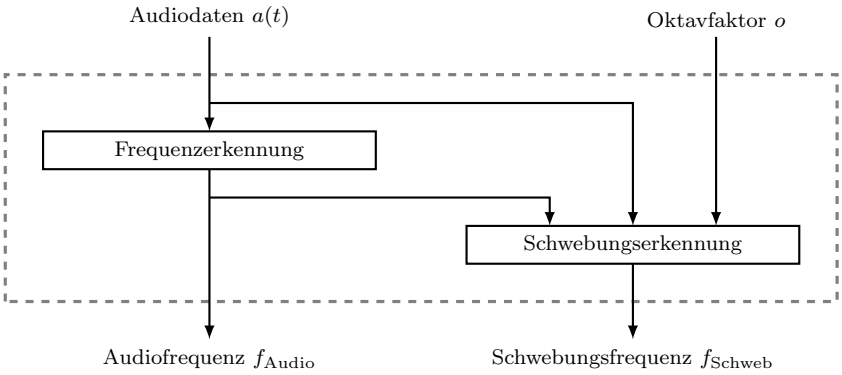


Abbildung 4.1: Datenflussdiagramm des entwickelten Schwebungserkennungsalgorithmus.

Basierend auf den gegebenen Audiodaten $a(t)$ wird die Frequenzerkennung durchgeführt und damit deren Frequenz gefunden. Für die Umsetzung können die verschiedenen, zuvor vorgestellten Algorithmen verwendet werden. Eine detailliertere Erklärung findet sich im folgenden Kapitel 4.2. In einem zweiten Schritt wird die Schwebungsfrequenz der Audiodaten ermittelt. Dabei werden die zuvor ermittelte Audiofrequenz f_{Audio} sowie der Oktavfaktor o der Ausgangstöne bzw. Orgelregister als zusätzliche Informationen verwendet. Diese werden, je nachdem welches Vorgehen zur Schwebungserkennung angewendet

wird (siehe Kapitel 4.3), als zusätzliche Informationen für die Berechnung benötigt. Als Ergebnis wird die aktuelle Schwebungsfrequenz f_{Schweb} zurückgegeben.

Die Berechnung wird jeweils auf Audiodaten mit einer Länge von 4 Sekunden durchgeführt. Damit ist bis zu einer minimalen Schwebungsfrequenz von 0,25 Hz jeweils mindestens eine vollständige Periode der Schwebung in diesen enthalten. Für die Ausführung der Frequenzerkennung ist prinzipiell auch eine geringere Fenstergröße ausreichend. Dort wird die Berechnung deshalb für verschiedene Abschnitte unabhängig voneinander ausgeführt und die einzelnen Ergebnisse zu einem einzigen Wert zusammengefasst.

4.2 Frequenzerkennung

Die Frequenzerkennung dient dazu zu erkennen, welche Frequenz in den gegeben Audiodaten aktuell vorliegt. Das dazugehörige Datenflussdiagramm ist in Abbildung 4.2 dargestellt.

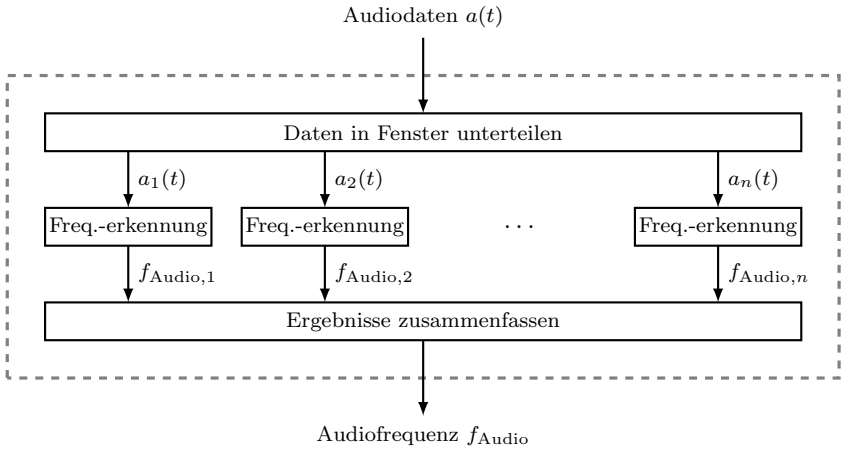


Abbildung 4.2: Datenflussdiagramm des Frequenzerkennungsalgorithmus.

Für die Frequenzerkennung werden die gegebenen Audiodaten in nebeneinanderliegende Fenster $a_1(t)$ bis $a_n(t)$ unterteilt. Bei den Testdaten mit einer Samplerate von 44,1 kHz wurde eine Fenstergröße W von 8.192 Samples gewählt. Dies entspricht ungefähr einer Zeitspanne von 186 ms. Die einzelnen Fenster werden so platziert, dass sich benachbarte Fenster zu jeweils 50 % überlappen. Somit ergibt sich eine Schrittweite von $\frac{W}{2}$ bzw. im konkreten Fall 4.096 Samples. Die Gesamtzahl der zu untersuchenden Fenster n kann ba-

sierend auf der Gesamtlänge der Audiodaten N und der Fenstergröße W folgendermaßen berechnet werden:

$$n = \left\lfloor \frac{N - W}{\frac{W}{2}} \right\rfloor + 1. \tag{4.1}$$

Auf alle so erzeugten Fenster wird die Frequenzerkennung angewendet. Dabei können die im vorherigen Kapitel vorgestellten Verfahren zum Einsatz kommen. Konkret sind dies: ZCR, ACF und SDF aus dem Zeitbereich sowie FFT, HPS, HSA und Cepstrum aus dem Frequenzbereich.

Die einzelnen Ergebnisse $f_{\text{Audio},1}$ bis $f_{\text{Audio},n}$ werden dann zu einem einzelnen Ergebnis zusammengefasst, indem ein hierarchisches Clustering durchgeführt wird. Alle Teilergebnisse, die ausreichend nah beieinander liegen, werden in gemeinsame Klassen einsortiert. Dafür wird das Distanzmaß D verwendet. Dieses berechnet sich jeweils als relativer Abstand der Durchschnittswerte zweier Cluster. Alle Cluster, deren Abstände zueinander kleiner als eine festgelegte Maximaldistanz D_{max} sind, werden während der Ausführung zusammengefasst. Detaillierte Informationen zum verwendeten Distanzmaß und zur Implementierung des Clustering-Algorithmus können in Kapitel A.1.1 im Anhang gefunden werden.

Für diesen Anwendungsfall wird für D_{max} ein Wert von 0,01 verwendet. Es werden also alle Werte zusammengefasst, deren relative Abstände zueinander kleiner als 1 % sind. Nach Abschluss des Clusterings wird der Durchschnitt der Klasse, die die meisten Elemente enthält, als finales Ergebnis f_{Audio} zurückgegeben.

4.3 Schwebungserkennung

Die Schwebungserkennung berechnet, basierend auf den Audiodaten $a(t)$, die darin vorliegende Schwebungsfrequenz f_{Schweb}. Dabei werden, je nach Verfahren, die Frequenz des Signals f_{Audio} oder der Oktavfaktor der Orgelregister o zusätzlich berücksichtigt. Für die durchgeführte Berechnung können verschiedene Ansätze gewählt werden. Die Erkennung kann basierend auf den Amplituden, der Hüllkurve oder den Fourierkoeffizienten des Signals erfolgen. Alle diese Möglichkeiten werden im Folgenden detailliert beschrieben.

4.3.1 Analyse der Amplituden

Bei diesem Verfahren wird, basierend auf den Amplituden des Signals, die Hüllkurve extrahiert und analysiert. Diese schwingt mit derselben Frequenz wie die gesuchte Schwebung. Das Datenflussdiagramm für dieses Verfahren ist in Abbildung 4.3 dargestellt.

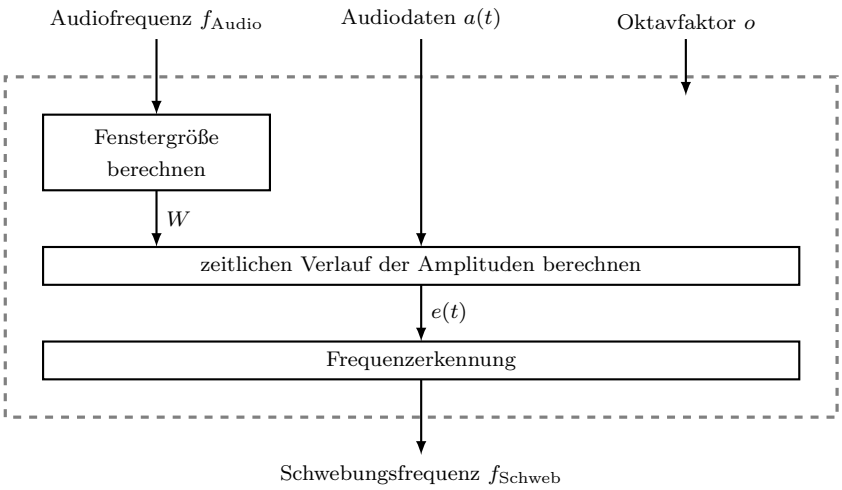


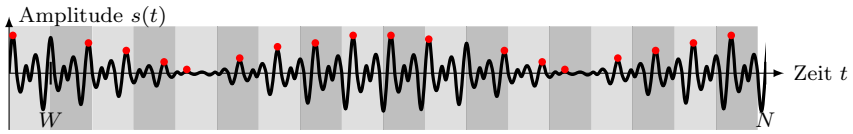
Abbildung 4.3: Datenflussdiagramm für die Schwebungserkennung durch Analyse der Amplituden.

Die Hüllkurve wird über den zeitlichen Verlauf der maximalen Amplituden der Daten berechnet. Für diese Berechnung wird eine Fenstergröße W verwendet, die sich wiederum aus der Frequenz f_{Audio} der Daten berechnet. Die resultierende Hüllkurve $e(t)$ kann dann durch einen der bekannten Frequenzerkennungsalgorithmen analysiert und somit die Schwebungsfrequenz f_{Schweb} ermittelt werden. Das verwendete Verfahren zur Ermittlung des zeitlichen Verlaufs der Amplitude ist in Abbildung 4.4 dargestellt.

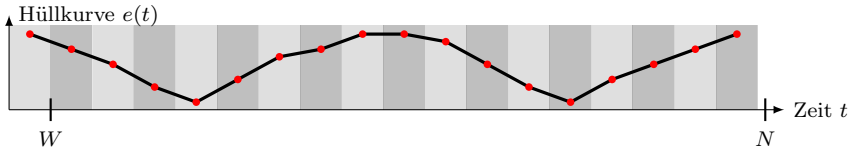
Die Ausgangsdaten $a(t)$ werden in benachbarte, nicht-überlappende Fenster gleicher Größe W eingeteilt (siehe Abbildung 4.4a). Für jedes der Fenster wird dessen globales Maximum berechnet. Die Aneinanderreihung $e(t)$ der einzelnen Maxima entspricht der Hüllkurve des Ausgangssignals (siehe Abbildung 4.4b).

Damit dieses Verfahren korrekt funktioniert, muss in jedem Fenster mindestens eine vollständige Periode des Signals enthalten sein. Nur dann ist sichergestellt, dass alle gefundenen Maxima auch jeweils die Maxima ihrer entsprechenden Periode sind. Dies ist vor allem bei niedrigen Frequenzen mit entsprechend langen Periodendauern kritisch. Deshalb muss die Frequenz der Audiodaten f_{Audio} bei der Berechnung der Fenstergröße W explizit berücksichtigt werden. Für diese sind die folgenden zwei Bedingungen festgelegt:

1. **W muss mindestens 1,1 Perioden der Audiodaten enthalten.** Damit wird sichergestellt, dass jedes der Fenster das Maximum mindestens einer Periode enthält.



(a) Ausgangssignal.



(b) Aus den Amplituden extrahierte Hüllkurve.

Abbildung 4.4: Vorgehen zur Ermittlung der Hüllkurve durch Analyse der Amplituden des Signals.

2. **W soll mindestens 2 % der Samplerate f_S betragen.** Damit wird erreicht, dass auch bei größer werdenden Frequenzen die Fenstergröße nicht unter eine Mindestgröße fällt.

Für die Berechnung der Fenstergröße W ist entsprechend eine Fallunterscheidung notwendig:

$$W = \begin{cases} 1{,}1 \cdot \dfrac{f_S}{f_{\text{Audio}}} & \text{falls } f \leq 55\,\text{Hz} \\ 0{,}02 \cdot f_S & \text{falls } f > 55\,\text{Hz} \end{cases}. \tag{4.2}$$

Bei Frequenzen bis zu 55 Hz kann W basierend auf der ersten Bedingung berechnet werden. Die zweite Bedingung ist in diesem Frequenzbereich immer erfüllt. Bei Frequenzen über 55 Hz ist dies hingegen nicht mehr der Fall und die zweite Bedingung wird verletzt:

$$W = 1{,}1 \cdot \frac{f_S}{55\,\text{Hz} + \epsilon} = \frac{1{,}1}{55\,\text{Hz} + \epsilon} \cdot f_S < 0{,}02 \cdot f_S. \tag{4.3}$$

In diesem Bereich wird deshalb dann die zweite Bedingung für die Berechnung der Fenstergröße W verwendet.

4.3.2 Analyse per Hüllkurvenerkennung

Dieses Verfahren arbeitet ähnlich zum vorher präsentieren Verfahren. Das dazugehörige Datenflussdiagramm ist in Abbildung 4.5 dargestellt. Die Extraktion der Hüllkurve $e(t)$ erfolgt hier nicht über die Amplituden, sondern durch ein spezielles Verfahren zur Erkennung der Hüllkurve. Danach wird diese ebenfalls über einen Frequenzerkennungsalgorithmus analysiert und somit die Schwebungsfrequenz f_{Schweb} berechnet.

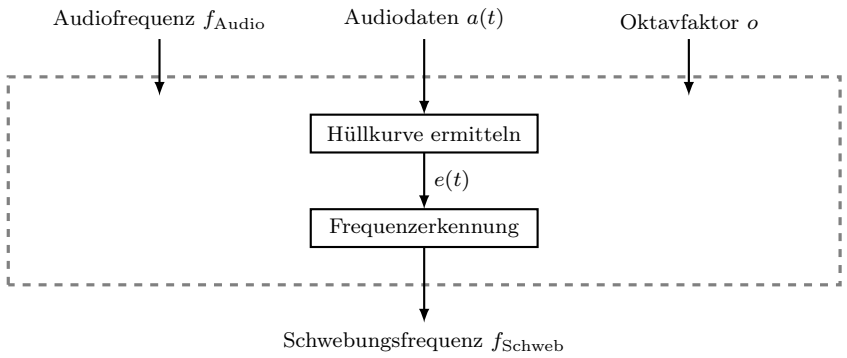


Abbildung 4.5: Datenflussdiagramm für die Schwebungserkennung durch Analyse per Hüllkurvenerkennung.

Der Vorteil dieses Ansatzes liegt darin, dass keine zusätzlichen Informationen benötigt werden. Die Audiodaten $a(t)$ sind für die Berechnung ausreichend. Das macht dieses Verfahren robuster gegenüber fehlerhaften Ergebnissen der Frequenzerkennung. Die Hüllkurvenerkennung verwendet das Verfahren, das in [Mat] beschrieben ist. Ein entsprechendes Datenflussdiagramm ist in Abbildung 4.6 dargestellt. Ein dazugehöriges Beispiel ist in Abbildung 4.7 gegeben.

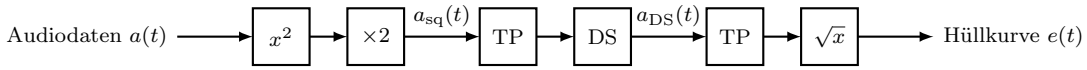
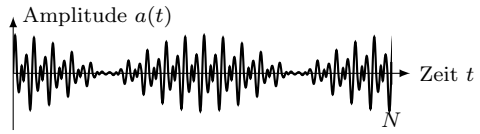


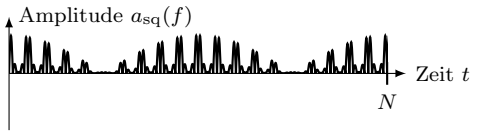
Abbildung 4.6: Datenflussdiagramm für die Hüllkurvenerkennung (TP: Tiefpassfilter, DS: Downsampling).

In einem ersten Schritt wird das Signal $a(t)$ quadriert und alle Werte verdoppelt (siehe Abbildung 4.7b). Das resultierende Signal $a_{\text{sq}}(t)$ durchläuft danach einen Downsampling-Schritt, der das Ausgangssignal auf die Zielsamplerate anpasst und das Signal $a_{\text{DS}}(t)$ liefert (siehe Abbildung 4.7c). Um dabei unerwünschte Frequenzen zu entfernen und Aliasing zu vermeiden, wird zuvor ein Tiefpassfilter angewendet. Die Extraktion der Hüllkurve $e(t)$ erfolgt im nächsten Schritt durch die erneute Anwendung eines Tiefpassfilters. Dadurch werden alle höherfrequenten Anteile des Signals entfernt und es bleibt nur die Hüllkurve übrig. Von dieser wird abschließend die Wurzel gezogen, um in den ursprünglichen Wertebereich zurückzukehren (siehe Abbildung 4.7d).

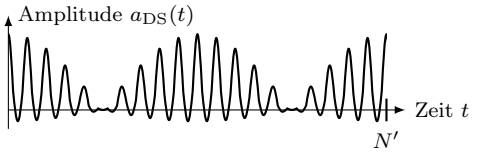
Für die Implementierung wurde eine Zielsamplerate von 100 Hz gewählt. Bei der verwendeten Fensterlänge von 4 Sekunden für die Schwebungserkennung stehen demnach mit insgesamt 400 Samples ausreichend Daten für eine genaue Berechnung zur Verfügung.



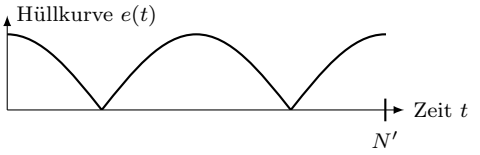
(a) Ausgangssignal $a(t)$.



(b) Demoduliertes Ausgangssignal $a_{\mathrm{sq}}(t)$.



(c) Signal $a_{\mathrm{DS}}(t)$ nach dem Anwenden des Tiefpassfilters und Downsampling.



(d) Extrahierte Hüllkurve $e(t)$.

Abbildung 4.7: Vorgehen zur Ermittlung der Hüllkurve des Signals per Hüllkurvenerkennung.

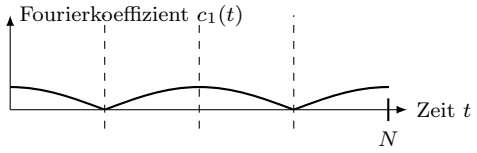
Nach dem Nyquist-Shannon-Abtasttheorem ergibt sich damit für den ersten Tiefpassfilter eine Cutoff-Frequenz von 50 Hz. Für den zweiten Tiefpassfilter wurde eine Cutoff-Frequenz von 10 Hz gewählt, so dass Schwebungsfrequenzen bis etwa zu dieser Frequenz gefunden werden können. Diese wurde gewählt, um sicherzustellen, dass keine hörbaren Frequenzen von über 15 Hz als Hüllkurve erkannt werden. Dies würde sonst, vor allem bei tieffrequenten Tönen, zu falschen Ergebnissen führen.

4.3.3 Analyse der Fourierkoeffizienten

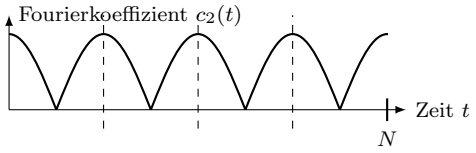
Wenn eine Schwebung auftritt, zeigt sich diese nicht nur in der Hüllkurve des Signals, sondern auch bei den Teiltönen. Immer wenn zwei Teiltöne annähernd gleicher Frequenz zusammen erklingen, entsteht eine Schwebung (vgl. [Her77], S. 300). Dabei entspricht die Schwebungsfrequenz ebenfalls der Differenz zwischen den beiden Ausgangsfrequenzen. Diese Schwebungen der einzelnen Teiltöne können im Amplitudenspektrum als zeitlich veränderliche Intensität der entsprechenden Teiltöne beobachtet werden, wie in Abbildung 4.8 dargestellt.

Bei dem Grundton entspricht die Schwingung des Fourierkoeffizienten $c_1(t)$ exakt der gesuchten Schwebungsfrequenz f_{Schweb} (siehe Abbildung 4.8b). Die Obertöne hingegen schwingen jeweils mit dem ganzzahligen Vielfachen der Schwebungsfrequenz. Grund dafür ist, dass eine Vervielfachung der beiden Ausgangsfrequenzen auch zu einer entsprechenden Vervielfachung deren Abstands und somit einer entsprechend höheren Schwebungsfre-

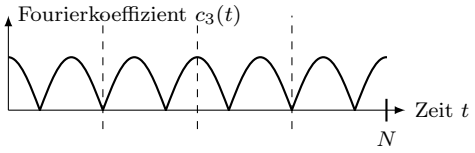
(a) Ausgangssignal $a(t)$



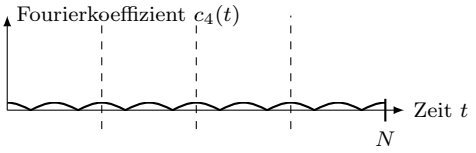
(b) Verlauf $c_1(t)$ der Fourierkoeffizienten des Grundtons.



(c) Verlauf $c_2(t)$ der Fourierkoeffizienten des ersten Obertons.



(d) Verlauf $c_3(t)$ der Fourierkoeffizienten des zweiten Obertons.



(e) Verlauf $c_4(t)$ der Fourierkoeffizienten des dritten Obertons.

Abbildung 4.8: Verläufe von Fourierkoeffizienten von Grund- und Obertönen für ein Signal mit vorliegender Schwebung.

quenz führt (siehe Abbildung 4.8c – 4.8e). So schwingt z.B. der erste Oberton doppelt so schnell wie der Grundton, der zweite Oberton dreimal so schnell, usw.

Dieser Zusammenhang kann auch für das Ermitteln der Schwebungsfrequenz, unter Zuhilfenahme von Kurzzeit-Fouriertransformationen, verwendet werden. Das entsprechende Datenflussdiagramm ist in Abbildung 4.9 gegeben. Für die Ausführung des Algorithmus wird zuvor eine Anzahl n_T von zu untersuchenden Tönen festlegt. Das bedeutet, es werden jeweils der Grundton und die ersten $n_T - 1$ Obertöne analysiert.

Im ersten Schritt wird die Fenstergröße W und die Schrittweite S für die Kurzzeit-Fouriertransformationen berechnet. Diese werden später verwendet um die Audiodaten $a(t)$ in Abschnitte zu unterteilen, auf die dann jeweils eine Fouriertransformation angewendet wird. Außerdem werden die Indizes der Bins berechnet, die dem Grundton und den zu untersuchenden Obertönen in den resultierenden Amplitudenspektren entsprechen. Dabei muss der Oktavfaktor der Register berücksichtigt werden. Die zu analysierenden Bins richten sich nach dem höherfrequenten Ton. Wenn z.B. zwei Töne von 100 Hz und 201 Hz gespielt werden, so liegt die Grundfrequenz ungefähr bei 100 Hz. Die Schwebungen im Amplitudenspektrum treten allerdings nur bei ungefähr 201 Hz, 402 Hz,

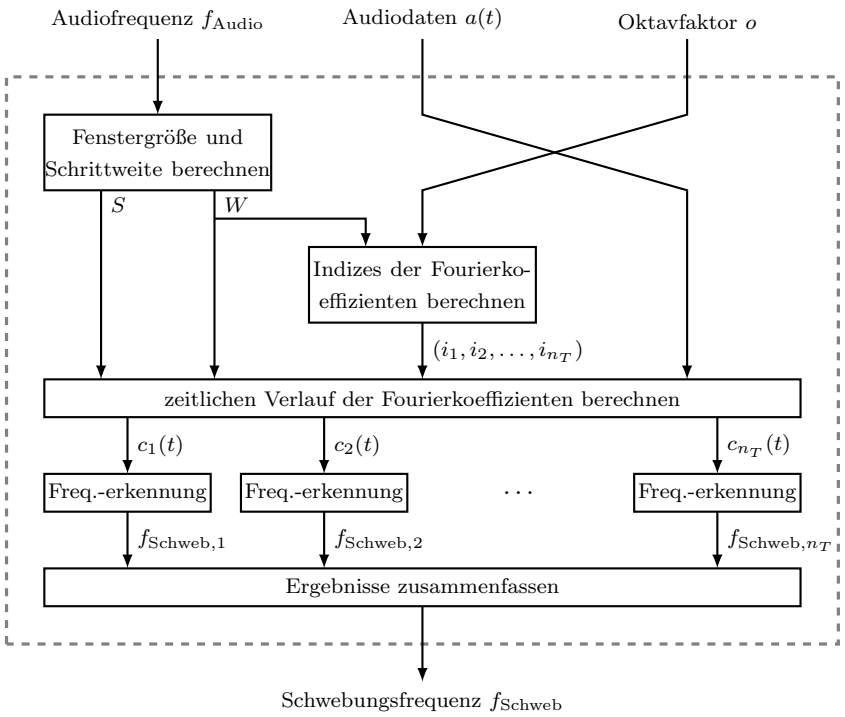


Abbildung 4.9: Datenflussdiagramm für die Schwebungserkennung durch Analyse der Fourierkoeffizienten.

usw. auf. Grund dafür ist, dass nur bei diesen Vielfachen die Grund- bzw. Obertöne der beiden Ausgangstöne aufeinander fallen. Bei anderen Frequenzen, wie z.B. 100 Hz oder 300 Hz, sind hingegen nur die Anteile des niederfrequenten Ausgangston vorhanden und es entsteht keine Schwebung.

Im nächsten Schritt wird der zeitliche Verlauf der Fourierkoeffizienten berechnet. Dazu werden die einzelnen Fouriertransformationen ausgeführt und die einzelnen Werte für jeden Koeffizienten in ein Array $c_i(t)$ gespeichert. Somit erhält man n_T Arrays, die den Verlauf der Koeffizienten enthalten. Jedes dieser Arrays wird im nächsten Schritt durch einen Frequenzerkennungsalgorithmus analysiert, wodurch sich wiederum n_T Schwebungsfrequenzen $f_{\text{Schweb},i}$ ergeben. Im letzten Schritt werden diese jeweils auf die Schwebungsfrequenz des Grundtons normiert (indem sie durch das entsprechende Vielfache geteilt werden) und durch einen Clustering-Algorithmus zu einem einzigen Ergebnis zusammengefasst. Hierzu wird derselbe Algorithmus, wie beim Zusammenfassen der einzelnen Ergebnisse der Frequenzerkennung (siehe Kapitel 4.2), verwendet. In diesem

Fall wird als Wert für die Maximaldistanz $D_{\max}$ 0,1 verwendet. Es werden also Ergebnisse zusammenfasst, deren Werte bis zu 10 % voneinander abweichen.

Die verwendete Fenstergröße W wird so gewählt, dass diese ungefähr 5 % der Samplerate f_S entspricht. Die Auflösung beträgt demnach ca. 20 Hz. Es werden also in jeden berechneten FFT-Bin ungefähr Frequenzen aus einem Bereich von 20 Hz einsortiert. Entsprechend können auch Schwebungsfrequenzen bis zu diesem Wert erkannt werden. Enthält die verwendete Fenstergröße nicht eine ganzzahlige Anzahl an Perioden des Ausgangssignals, so entstehen bei der Fouriertransformation durch den Leck-Effekt Ungenauigkeiten, indem sich benachbarte Koeffizienten gegenseitig beeinflussen (vgl. [SS11], S. 783 f.). Um diese möglichst gering zu halten, wird der exakte Wert der Fenstergröße W so gewählt, dass eine ganzzahlige Anzahl von Perioden des Ausgangssignals enthalten ist. Mithilfe der gegebenen Frequenz f_{Audio} wird die Fenstergröße W auf die nächstgrößere Periode angepasst. Die Anzahl der zu betrachtenden Perioden n_P kann folgendermaßen berechnet werden:

$$n_P = \lceil 0{,}05 \cdot f_{\text{Audio}} \rceil \,. \tag{4.4}$$

Mit dieser Periodenanzahl kann die Fenstergröße W, basierend auf der Audiofrequenz f_{Audio} und der Samplerate f_S, berechnet werden:

$$W = \left\lceil \frac{n_P}{f_{\text{Audio}}} \cdot f_S \right\rceil . \tag{4.5}$$

Die Schrittweite S wird so gewählt, dass eine gewünschte Zeitgenauigkeit g erreicht wird. Für das entwickelte System wurde eine Genauigkeit von $g = 0{,}01\,\text{s}$ verwendet, so dass jede Sekunde der Audiodaten $a(t)$ zu 100 Ergebniswerten wird. Der genaue Wert von S kann über die folgende Formel berechnet werden:

$$S = g \cdot f_S. \tag{4.6}$$

4.3.4 Grundlegende Überprüfung auf das Vorhandensein einer Schwebung

Die oben beschriebenen Vorgehen verwenden eine Frequenzerkennung, um die vorliegende Schwebungsfrequenz zu ermitteln. Zusätzlich muss jedoch auch der Fall erkannt werden, wenn im zu analysierenden Audiosignal keine Schwebung vorhanden ist. Diese Überprüfung muss für die untersuchten Eingabesignale $e(t)$ (also jeden Amplitudenverlauf und jede Hüllkurve) und $c_i(t)$ (also jeden Verlauf eines Fourierkoeffizienten) durchgeführt werden. Dazu wird das Verhältnis v zwischen dem globalen Minimum und dem globalen Maximum gebildet:

$$v = \frac{\min(e(t))}{\max(e(t))} \quad \in [0, 1] \text{ bzw.} \tag{4.7}$$

$$v = \frac{\min(c_i(t))}{\max(c_i(t))} \quad \in [0, 1]. \tag{4.8}$$

Enthalten die Daten keine Amplituden (und es liegt dementsprechend keine Schwebung vor), so ist der Unterschied zwischen dem Minimum und Maximum relativ gering und der Wert v nah am Maximalwert 1. Sind hingegen Schwebungen enthalten, so ist der Abstand der Werte größer und v entsprechend kleiner. Um zu bestimmen, ob eine Schwebung vorliegt oder nicht, wird v mit einem zuvor festgelegten Parameter V_{nobeat} verglichen:

$$\begin{cases} \text{keine Schwebung vorhanden} & \text{falls } v \geq V_{\text{nobeat}} \\ \text{Schwebung vorhanden} & \text{falls } v < V_{\text{nobeat}} \end{cases}. \tag{4.9}$$

Ist v größer als oder gleich groß wie der Parameter V_{nobeat}, so kann davon ausgegangen werden, dass keine Schwebung vorliegt. Das Ergebnis der Schwebungserkennung kann folglich auf 0 Hz gesetzt werden. Im entgegengesetzten Fall ist wahrscheinlich eine Schwebung vorhanden und eine weitere Analyse von $e(t)$ bzw. $c_i(t)$ durch den jeweiligen Frequenzerkennungsalgorithmus notwendig.

4.4 Implementierung der verwendeten Algorithmen

Detaillierte Erklärungen über die Implementierung der verwendeten Algorithmen finden sich im Anhang A.1. Dort werden der Clustering-Algorithmus (in Kapitel A.1.1) und der Interpolationsalgorithmus zur Verbesserung der Genauigkeit der FFT-Ergebnisse (in Kapitel A.1.2) erläutert. In Kapitel A.1.3 findet sich ein Überblick über die Implementierung der verwendeten Frequenzerkennungsverfahren.

4.5 Softwarearchitektur

Die Umsetzung des Schwebungserkennungsalgorithmus ist als objektorientierte Implementierung in Python erfolgt. Ein dazugehöriges Klassendiagramm ist in Abbildung 4.10 gegeben.

Der entwickelte Schwebungserkennungsalgorithmus wird in der Klasse `OverallSystem` umgesetzt. Über deren Funktion `Process()` werden die Audiodaten übergeben und die Berechnung ausgeführt. Die resultierenden Ergebnisse der Frequenz- und Schwebungserkennung können über die Attribute `CurrentPitch` bzw. `CurrentBeat` ausgelesen werden. Über weitere Attribute kann die Konfiguration des Systems, z.B. die verwendeten Fenstergrößen, festgelegt werden. Jeder Instanz dieser Klasse ist je eine Komponente zur Frequenz- und Schwebungserkennung zugeteilt. In diesen erfolgt dann die Berechnung.

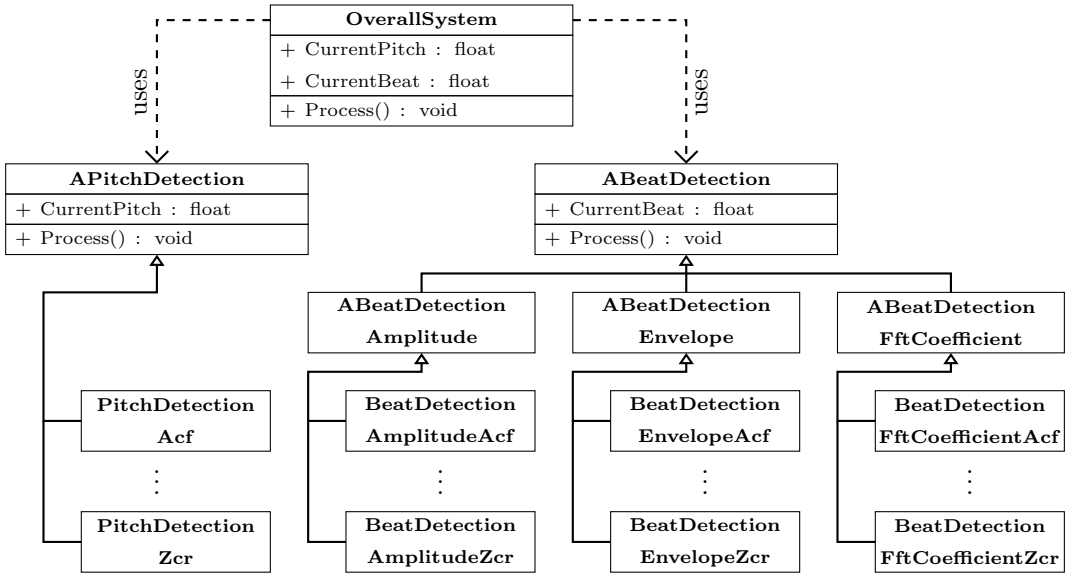


Abbildung 4.10: Klassendiagramm der Python-Implementierung des Gesamtsystems.

Alle Frequenzerkennungen werden über die abstrakte Klasse `APitchDetection` modelliert. Diese definiert die Funktionen, die zur grundlegenden Ausführung benötigt werden. Hier existiert, wie bei `OverallSystem`, eine `Process()`-Funktion, die für gegebene Audiodaten deren Frequenz berechnet. Über das Attribut `CurrentPitch` kann diese ausgelesen werden. Die einzelnen Implementierungen der Frequenzerkennung erfolgen in den dazugehörigen Kindklassen. Dabei existiert für jeden verwendeten Algorithmus, also ACF, Cepstrum, FFT, HPS, HSA, SDF und ZCR, eine gesonderte Klasse.

Bei der Schwebungserkennung existiert für jeden der möglichen Ansätze eine zusätzliche abstrakte Elternklasse: `ABeatDetectionAmplitude` für die Analyse der Amplituden, `ABeatDetectionEnvelope` für die Analyse der Hüllkurve und `ABeatDetectionFftCoefficient` für die Analyse der Fourierkoeffizienten. Jede dieser Klassen enthält die grundlegende Funktionalität zur Berechnung der jeweils betrachteten Eigenschaften. Die Analyse dieser, und damit die Berechnung der Schwebungsfrequenz, erfolgt über die Implementierung in den Kindklassen.

4.6 Testumgebung

Für die Tests des entwickelten Algorithmus wurde eine passende Testumgebung implementiert. Mithilfe dieser können sowohl die Algorithmen zur Frequenz- und Schwebungserkennung einzeln als auch das Gesamtsystem getestet werden.

Der prinzipielle Aufbau der Testdaten ist in der Abbildung 4.11 dargestellt. Die verwendeten Testdaten sind in Testsets organisiert. Jedes Testsets besteht aus verschiedenen Daten, die sich z.B. nach Größe oder Ursprung grundlegend unterscheiden können. Innerhalb der Testsets sind die Daten als Testsuites strukturiert. Eine Testsuite enthält vom Aufbau her ähnliche Testdaten, wie sie z.B. durch die Verwendung einer bestimmten Kombination von Labial- und Lingualregister entstehen. Zusätzlich zu den Audiodateien enthält jede Testsuite eine JSON-Datei (JavaScript Object Notation), in der die erwarteten Audio- und Schwebungsfrequenzen gespeichert sind. Diese ermöglicht eine automatisierte Ausführung und Auswertung des Testsets.

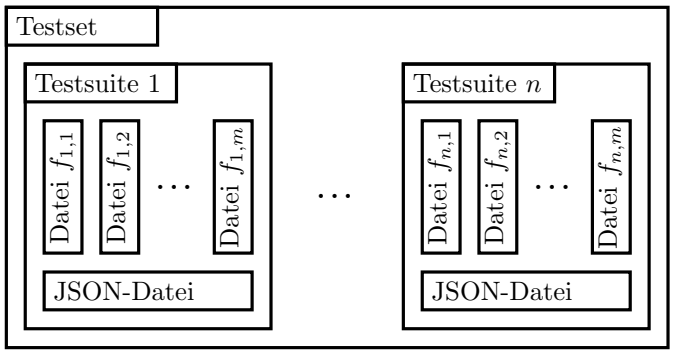


Abbildung 4.11: Aufbau eines Testsets.

Für die Ausführung der Tests einer einzelnen Implementierung werden nacheinander alle Testsuites des betrachteten Testsets verarbeitet. Dazu werden alle enthaltenen Audiodateien geladen, in festgelegte Fenster unterteilt und diese als Eingabe für die Algorithmen verwendet. Die einzelnen Ergebnisse jeder Testsuite werden zusammen in einer CSV-Datei gespeichert. Diese enthält Informationen zu den analysierten Dateien, den untersuchten Fenstern sowie den jeweils erwarteten und gefundenen Ergebnissen. Diese Rohergebnisse alles Testsuites eines Testsets werden in einem finalen Schritt zusammengefasst. Dazu werden die erstellten CSV-Dateien wieder eingelesen und untersucht. Die finalen Ergebnisse werden wiederum in einer einzigen CSV-Datei gespeichert. Diese enthält dann u.a. Informationen über die resultierenden Trefferquoten, Genauigkeiten und durchschnittlichen Abweichungen.

Die Implementierung der Testumgebung erfolgte ebenfalls in Python. Um die Ausführung der Tests so weit wie möglich zu beschleunigen, werden mehrere Testdateien der jeweils aktuellen Testsuite auf den zur Verfügung stehenden Prozessorkernen parallel verarbeitet. Dazu wird die `multiprocessing`-Bibliothek verwendet. Im Gegensatz zur `threading`-Bibliothek hat diese den Vorteil, dass für die einzelnen Threads Subprozesse generiert werden und somit das Global Interpreter Lock umgangen wird (vgl. [Pyt22]). Dieses sorgt sonst dafür, dass zu jedem Zeitpunkt nur exakt ein Thread auf den Python-Interpreter zugreifen kann. Das würde wiederum dazu führen, dass nicht alle zur Verfügung stehenden Kerne ausgelastet werden.

4.7 Programmierumgebung

Für die Entwicklung des Python-Codes wurde die integrierte Entwicklungsumgebung (Integrated development environment, IDE) *Spyder* in der Version 5.1.5 zusammen mit *Python 3.9.7 64-bit* verwendet. Zur Versionskontrolle wurde ein Git-Repository im GitLab der TU Chemnitz verwendet[1]. Bei der Implementierung wurden an verschiedenen Stellen zusätzliche Bibliotheken verwendet.

Für das Gesamtsystem kam ausschließlich die `numpy`-Bibliothek zum Einsatz (Version 1.20.3). Diese stellt verschiedene häufig verwendete mathematische Funktionen zur Verarbeitung von Vektoren, Arrays und Matrizen bereit. Verwendet werden davon u.a. `numpy.log()` zur elementweisen Berechnung des Logarithmus eines Arrays und `numpy.inner()` zur Berechnung des Skalarprodukts zweier Vektoren. Zusätzlich bietet sie verschiedene Funktionen, die zur Signalverarbeitung verwendet werden. `numpy.blackman()` und `numpy.hanning()` dienen zur Anwendung der Blackman- bzw. Hanning-Fensterfunktion auf gegebene Daten. Die Funktion `numpy.fft.rfft()` wird zur Berechnung des Amplitudenspektrums verwendet.

Bei der Implementierung der Testumgebung kamen zusätzlich die folgenden Bibliotheken zum Einsatz:

- Die `json`-Bibliothek in der Version 2.0.9: diese wird zum Einlesen der Definitionsdateien für die Testdaten, die im JSON-Format vorliegen, verwendet.
- Die `csv`-Bibliothek in der Version 1.0 und die `statistics`-Bibliothek: diese werden zum Speichern, Einlesen und Auswerten der Testergebnisse verwendet.
- Die Audio-Bibliothek `librosa` in der Version 0.9.2: diese bietet mit der Funktion `librosa.load()` eine Möglichkeit Audiodateien in Python zu laden.

[1] `https://gitlab.hrz.tu-chemnitz.de/nijun--tu-chemnitz.de/ma-stimmgeraet-orgelpfeifen`

Für die Erstellung der Testdaten wurden zusätzlich die folgenden Bibliotheken verwendet:

- Die `midiutil`-Bibliothek in der Version 1.2.1: diese wird verwendet um die MIDI-Dateien (Musical Instrument Digital Interface) für die Erstellung der Testdaten zu generieren.
- Die `pyrubberband`-Bibliothek in der Version 0.3.0: Diese wird als Wrapper für das Kommandozeilentool der Audiobibliothek Rubberband verwendet. Diese muss zusätzlich installiert sein. Hier wurde diese in der Version 3.0.0 verwendet.

5 Generierung der Testdaten

Dieses Kapitel gibt einen Überblick über die verwendeten Testdaten. Dabei wird auf deren Aufbau (Kapitel 5.1) und Erzeugung (Kapitel 5.2) eingegangen. Abschließend wird ein Überblick über alle verwendeten Testsets gegeben (Kapitel 5.3).

5.1 Aufbau der Testdaten

Für die Tests des entwickelten Algorithmus kommen verschiedene Testsets zum Einsatz. Diese unterscheiden sich hinsichtlich ihrer Größe, den vorkommenden Schwebungsfrequenzen und des verwendeten Syntheseansatzes (additive Synthese oder GrandOrgue-basierte Synthese). Jede enthaltene Testsuite entspricht der Paarung von je einem Labial- und Lingualregister und enthält folglich alle Testdaten für diese Paarung.

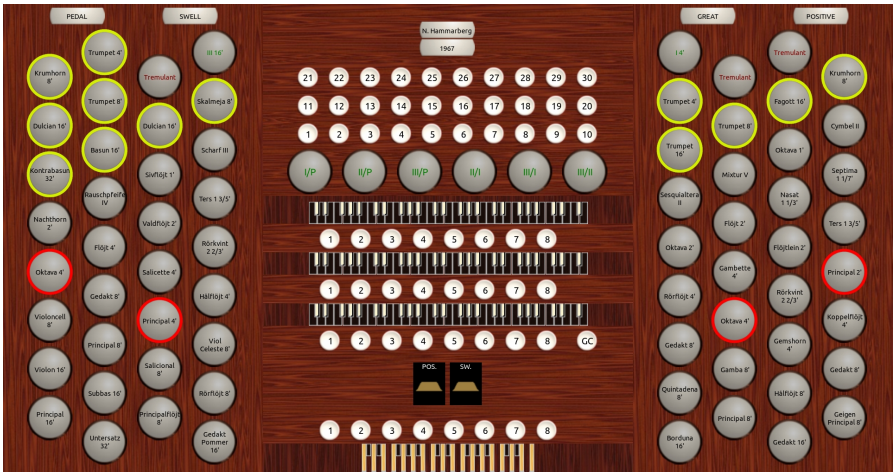


Abbildung 5.1: Bureå Church (extended AGO version)-Orgel in GrandOrgue. Markiert sind die verwendeten Labialregister (rot) und Lingualregister (gelb).

Die einzelnen Testsuites orientieren sich am Aufbau der *Bureå Church (Extended AGO version)*-Orgel für den kostenfreien Orgelsimulator *GrandOrgue* (siehe [LG]). Ihr Aufbau ist in Abbildung 5.1 dargestellt. Dieser wurde für die Erzeugung der additiven und GrandOrgue-basierten Testdaten verwendet. Insgesamt können 13 Paarungen von

Labial- und Lingualregistern erstellt werden, die in Tabelle 5.1 gegeben sind. Dort sind für jedes Werk das als Referenz zu verwendende Labialregister sowie alle zu stimmenden Lingualregister aufgeführt.

Werk	Hauptwerk	Schwellwerk	Positiv	Pedal
Labial-register	Oktave 4’	Prinzipal 4’	Prinzipal 2’	Oktava 4’
Lingual-register	Trompete 4’	Schalmei 8’	Krummhorn 8’	Trompete 4’
	Trompete 8’	Dulcian 16’	Fagott 16’	Trompete 8’
	Trompete 16’			Krummhorn 8’
				Posaune 16’
				Dulcian 16’
				Posaune 32’

Tabelle 5.1: Übersicht über die vorhandenen Labial- und Lingualregistern der verwendeten Bureå Church (extended AGO version)-Orgel.

Jede in einer Testsuite enthaltenen Datei hat eine Länge von 10 Sekunden und liegt als WAV-Dateien im 32 bit float-Format vor. Für jede dieser Dateien ist die zu erkennende Audio- und Schwebungsfrequenz in der zur Testsuite gehörenden JSON-Datei hinterlegt.

Um möglichst umfangreiche Testdaten zu haben, liegen diese für jeden möglichen Ton der Manuale bzw. des Pedals mit den 13 folgenden Frequenzabweichungen vor:

- $-5{,}0\,\mathrm{Hz}$
- $-2{,}0\,\mathrm{Hz}$
- $-1{,}0\,\mathrm{Hz}$
- $-0{,}5\,\mathrm{Hz}$
- $-0{,}2\,\mathrm{Hz}$
- $-0{,}1\,\mathrm{Hz}$
- $\pm 0{,}0\,\mathrm{Hz}$
- $+0{,}1\,\mathrm{Hz}$
- $+0{,}2\,\mathrm{Hz}$
- $+0{,}5\,\mathrm{Hz}$
- $+1{,}0\,\mathrm{Hz}$
- $+2{,}0\,\mathrm{Hz}$
- $+5{,}0\,\mathrm{Hz}$.

Bei tiefen Frequenzen kann es auftreten, dass durch das Mischen der beiden Ausgangstöne keine Schwebung entsteht. Die Ursache dafür ist, dass dort die Frequenzabstände zwischen benachbarten Tönen deutlich geringer sind und sich dadurch keine oder nur schwer erkennbare Schwebungen bilden. Dieses Problem tritt am deutlichsten auf, wenn die Ausgangstöne in unterschiedlichen Oktaven gespielt werden. Mischt man z.B. die beiden Frequenzen $f_1 = 120\,\mathrm{Hz}$ und $f_2 = 32\,\mathrm{Hz}$ (was einer Abweichung von $2\,\mathrm{Hz}$ zum

erwarteten Wert $f_2 = 30\,\text{Hz}$ entspricht), müsste rechnerisch eine Schwebungsfrequenz von 8 Hz entstehen. Im resultierenden Signal ist diese allerdings nicht beobachtbar, weil dieses durch die Schwingung des niederfrequenten Tons dominiert wird (siehe Abbildung 5.2). Um solche Fälle bei den Tests auszuschließen, wird eine obere Schranke $f_{\text{Schweb,max}}$ für die zum Test zugelassenen Schwebungsfrequenzen berechnet:

$$f_{\text{Schweb,max}} = 0{,}2 \cdot \frac{f_1}{o}, \qquad f_1 > f_2. \tag{5.1}$$

Eine Testdatei wird also nur akzeptiert, wenn deren Schwebungsfrequenz kleiner als 20 % der Frequenz des tieferfrequenten Tons (diese berechnet sich mithilfe der Frequenz des höherfrequenten Tons f_1 und dem Oktavfaktor o) ist. Für alle anderen Fälle werden keine Dateien erstellt und diese somit auch nicht für die Tests verwendet. Bei Tönen, die in derselben Oktave angespielt werden, entspricht der Faktor 0,2 einem Tonabstand von knapp 4 HTS[1]. Da Schwebungen bei einem Abstand von mehr als 3 HTS undeutlich werden (vgl. [Her77], S. 299), ist dieser Faktor ausreichend groß gewählt.

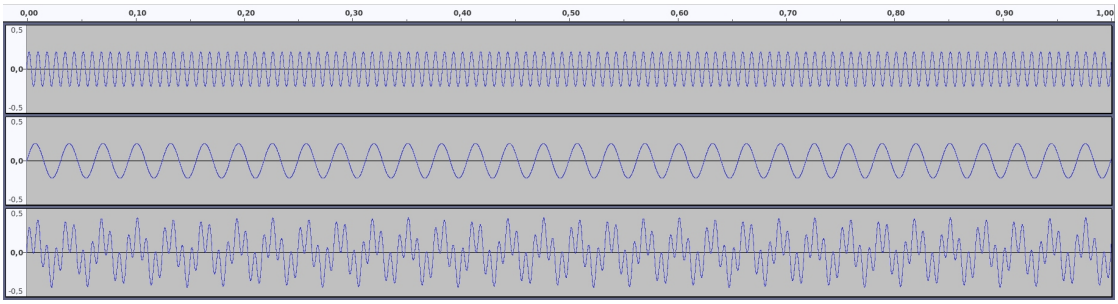


Abbildung 5.2: Die Überlagerung von zwei Schwingungen mit den Frequenzen 120 Hz (oben) und 32 Hz (Mitte) führt zu keiner sichtbaren Schwebung (unten).

Bei den 7 Registerpaarungen der Manuale (Hauptwerk, Schwellwerk, Positiv), mit je 56 Tasten, können jeweils bis zu 728 Testdateien erstellt werden. Für die 6 Registerpaarungen im Pedal mit seinen 30 Tasten jeweils bis zu 390 Dateien. Somit ergibt sich eine Gesamtanzahl von bis zu 7.436 Testdateien pro Testset.

Da ein solches komplettes Testset mit einer Gesamtlänge von ca. 20,7 Stunden relativ lang ist, werden von allen Testsets auch reduzierte Varianten erstellt. Somit können zeit-

[1] Wenn f_1 die höherfrequente Schwingung ist, gilt für die Frequenz f_2:

$$f_2 = f_1 - 0{,}2 \cdot f_1 = 0{,}8 \cdot f_1. \tag{5.2}$$

Der Faktor 0,8 entspricht nach Formel 2.5 einem Abstand von -3,86 HTS.

aufwendige Ausführungen, wie z.B. zum Finden der optimalen Parameter, beschleunigt werden. Um dies zu erreichen, wird sowohl die Anzahl der Töne als auch der vorhandenen Schwebungsfrequenzen reduziert. Jede der Testsuiten enthält dann nur noch 12 Töne. Diese werden so ausgewählt,

- dass jeder der 12 Halbtöne der Oktave genau einmal vorkommt und
- dass diese möglichst gleichmäßig über die komplette Klaviatur verteilt sind.

Da es für die wahrnehmbare Schwebungsfrequenz keinen Unterschied macht, ob der vorhandene Frequenzunterschied positiv oder negativ ist, reicht es auch aus eine der beiden Frequenzabweichungen zu behalten und diese somit zu halbieren. Zusätzlich werden die maximalen (±5,0 Hz) und minimalen (±0,1 Hz) Abweichungen entfernt. In den resultierenden Testsuiten sind dann also nur noch die folgenden 5 Abweichungen enthalten:

- $-2{,}0\,\mathrm{Hz}$
- $-0{,}5\,\mathrm{Hz}$
- $\pm 0{,}0\,\mathrm{Hz}$
- $+0{,}2\,\mathrm{Hz}$
- $+1{,}0\,\mathrm{Hz}$.

Somit sinkt die Anzahl der Testdateien pro Testsuite auf bis zu 60 Dateien. Das entspricht einer Verringerung von ca. 91,8 % bzw. 84,6 %. Die Größe des kompletten Testsets verringert sich dadurch auch auf maximal 780 Testdateien und somit ca. 10,5 % der ursprünglichen Größe.

5.2 Erzeugung der Testdaten

Für die Generierung der Testdaten kommen zwei verschiedene Ansätze zum Einsatz. Diese sind:

- Generierung per additiver Synthese und
- Generierung mithilfe des GrandOrgue-Orgelsimulators.

Allgemeine Informationen zum Erstellen des Testdaten finden sich im Kapitel 5.2.1. In den darauffolgenden Kapiteln 5.2.2 und 5.2.3 wird das spezifische Vorgehen für die beiden verwendeten Ansätze genauer erklärt.

5.2.1 Allgemeines Vorgehen

Das grundlegende Vorgehen zum Erstellen der Testdaten ist in Abbildung 5.3 gegeben. Für Erstellung der Testdaten einer Registerpaarung müssen deren Audiodaten jeweils als

eine Datei übergeben werden. Diese sind so aufgebaut, dass sie alle Töne des Registers nacheinander und in aufsteigender Reihenfolge enthalten. Die Länge jedes Tons entspricht dabei exakt 16 Sekunden, aufgeteilt auf 12 Sekunden Ton und 4 Sekunden Stille. Das macht ein automatisiertes Einlesen und Trennen der einzelnen Töne möglich.

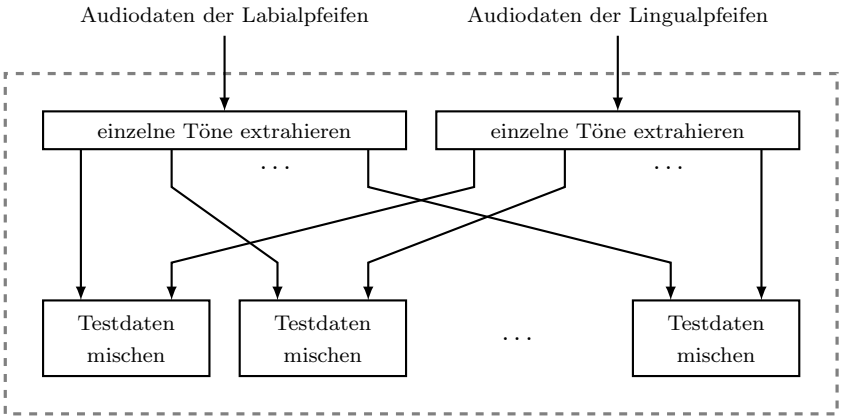


Abbildung 5.3: Datenflussdiagramm für die Erzeugung der Testdaten aus Audioaufnahmen von zwei Orgelregistern. Der Block *Testdaten mischen* ist in Abbildung 5.4 detaillierter dargestellt.

Für jede Testsuite werden schrittweise die zueinander gehörigen Töne der Labial- und Lingualregister gemischt und gespeichert. Dabei wird jeweils die Frequenz des Tons der Lingualpfeife entsprechend der gewünschten Abweichung verändert. Um dies umzusetzen, wird ein Pitch-Shifting-Algorithmus angewendet. Dabei kann zwischen den folgenden zwei Optionen gewählt werden:

- Die Funktion `librosa.effects.pitch_shift` der librosa-Audiobibliothek für Python ([lib13]).
- Die Funktion `pyrubberband.pyrb.pitch_shift` der pyrubberband-Audiobibliothek für Python ([Bri15]).

Durch die Verwendung zwei verschiedener Implementierungen kann später untersucht werden, ob diese einen Einfluss auf die Ergebnisse haben und diese verfälschen. Die resultierenden Audiodaten werden auf eine Länge von 10 Sekunden gekürzt als WAV-Datei im 32 bit float-Format gespeichert. Das hat den Vorteil, dass vor dem Speichern die Notwendigkeit zur Durchführung von Dithering entfällt. Diese Schritte werden für alle Töne und Frequenzabweichungen wiederholt.

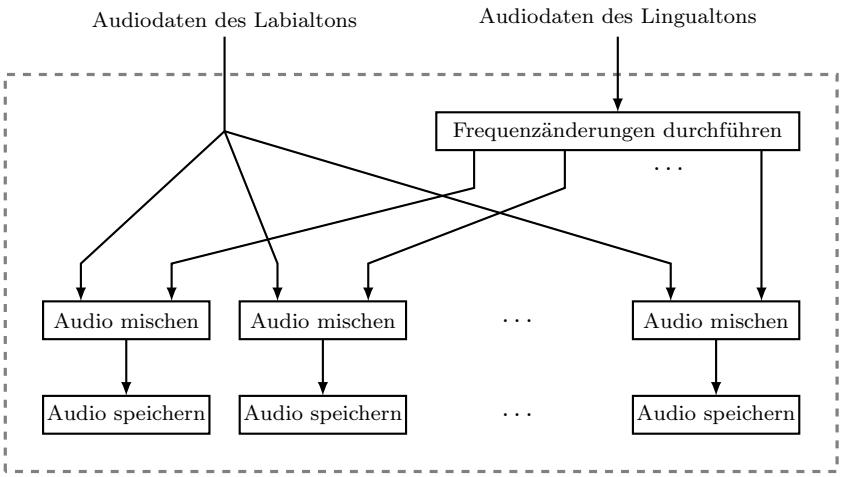


Abbildung 5.4: Datenflussdiagramm zum Mischen der Testdaten aus zwei einzelnen Tönen.

Zusätzlich zu den Audiodaten werden auch zwei JSON-Dateien für die Testsuite erstellt. Diese enthalten jeweils den Namen und die Beschreibung der Testsuite sowie Informationen zu allen enthaltenen Testfällen. Für jeden Testfall werden dabei die folgenden Informationen gespeichert:

- relativer Pfad zur WAV-Datei ausgehend von der JSON-Datei,
- in der Datei vorhandene Audiofrequenz und
- in der Datei vorhandene Schwebungsfrequenz.

Eine der beiden JSON-Dateien enthält Informationen zu allen Testfällen. Die zweite Datei spiegelt jeweils die reduzierte Testsuite wider und enthält demnach entsprechend weniger Informationen.

5.2.2 Additive Synthese

Bei diesem Ansatz wird eine additive Synthese verwendet, um die Ausgangsdaten zu generieren, die später für die Erstellung der Testdaten verwendet werden. Dazu wurden die Spektren der Töne im mittleren Bereich jedes Orgelregisters des GrandOrgue-Orgelsimulators (ebenfalls mit der Bureå Church (Extended AGO version)-Orgel) analysiert. Bei den Manualregistern mit ihren 56 Tasten war dies der 28. Ton, bei den Pedalregistern mit 30 Tasten der 15. Ton.

Die damit ermittelten Frequenzanteile des Grundtons und der Obertöne werden für die Synthese der Töne verwendet. Dabei werden jeweils 12 Sekunden lange Töne für alle benötigten Frequenzen generiert und mit einem Abstand bzw. Stille von 4 Sekunden aneinandergefügt. Das resultierende Signal wird abgespeichert und kann dann für die Erstellung der Testdaten verwendet werden. Dieses Vorgehen wird für alle 17 benötigten Register wiederholt.

Diese Testdaten können vor allem für grundlegende Tests verwendet werden, da sie keine Störgeräusche, Rauschen, Hall o.ä. enthalten.

5.2.3 GrandOrgue-basierte Synthese

Der Hauptvorteil bei der Verwendung von GrandOrgue für die Erzeugung der Testdaten liegt darin, dass es sich um einen samplebasierten Orgelsimulator handelt. Das bedeutet, dass die resultierenden Klänge aus einzelnen Samples realer Orgeln zusammengesetzt sind. Das führt dazu, dass die Audiodaten von der Komplexität her eine hohe Ähnlichkeit zu realen Aufnahmen aufweisen.

Für die Erstellung der Testdaten mit dem GrandOrgue-Orgelsimulator müssen die Audiodaten in der erwarteten Struktur vorliegen (pro Ton jeweils 12 Sekunden Audio gefolgt von 4 Sekunden Stille). Um diese zu erstellen, kann die Aufnahmefunktion in GrandOrgue verwendet werden. Mit dieser kann direkt eine MIDI-Datei abgespielt sowie das resultierende Audio aufgenommen und gespeichert werden. Die MIDI-Datei enthält dabei alle Steuerbefehle, damit die richtigen Töne zum richtigen Zeitpunkt gespielt werden. Indem dieses Vorgehen für alle benötigten Register wiederholt wird, können die Ausgangsdaten für die Erstellung der Testdaten generiert werden.

Die benötigten MIDI-Dateien wurden mithilfe eines entsprechendes Python-Skripts, unter Zuhilfenahme der `midiutil`-Bibliothek, erstellt. Dabei wurden insgesamt 4 Dateien erstellt, je eine pro Werk (Pedal, Hauptwerk, Schwellwerk, Positiv).

Für die Verwendung der GrandOrgue-basierten Synthese ist zusätzlich noch eine Anpassung bei der Erstellung der Testdaten notwendig. Die Frequenzverhältnisse der realen Orgelklänge weichen zu einem gewissen, wenn auch geringen, Grad von dem mathematischen Modell der gleichstufigen Stimmung (siehe Gleichung 2.5) ab. Für die Erstellung der Testdaten ist es allerdings notwendig, die exakten Frequenzen der beiden Ausgangstöne zu kennen. Sonst kann es dazu kommen, dass die resultierenden Audio- und Schwebungsfrequenzen nicht mit den theoretisch berechneten Werten übereinstimmen. Das wiederum führt zu Fehlern bei der automatisierten Testdurchführung. Deshalb werden zusätzlich zu den Audiodaten auch die exakten Frequenzen aller Töne mit übergeben, um dieses Problem bei der Erstellung zu vermeiden.

5.3 Überblick über die verwendeten Testsets

Ein tabellarischer Überblick über alle erzeugten und in der Evaluation verwendeten Testsets ist in Tabelle 5.2 gegeben.

Nr.	Name	Ausgangsdaten	Pitch-Shifting	Größe
1	Additive 1 complete	additiv	librosa	vollständig
2	Additive 1 reduced	additiv	librosa	reduziert
3	Additive 2 complete	additiv	pyrubberband	vollständig
4	Additive 2 reduced	additiv	pyrubberband	reduziert
5	GrandOrgue 1 complete	GrandOrgue	librosa	vollständig
6	GrandOrgue 1 reduced	GrandOrgue	librosa	reduziert
7	GrandOrgue 2 complete	GrandOrgue	pyrubberband	vollständig
8	GrandOrgue 2 reduced	GrandOrgue	pyrubberband	reduziert

Tabelle 5.2: Überblick über verwendete Testsets.

Sämtliche generierten Testdaten liegen als WAV-Dateien mit jeweils einem Audiokanal (mono) im 32 bit float-Format vor. Jedes vollständige Testset besteht aus 6.968 Audiodateien, die insgesamt ca. 12,3 GB Speicherplatz belegen. Die Größe jedes reduzierten Testsets würde rechnerisch ca. 1,3 GB betragen. Da jedoch die Audiodaten der vollständigen Testsets ohne Einschränkung wiederverwendet werden können, fällt kaum zusätzlicher Speicherplatz an (lediglich weniger als 200 kB für die benötigten JSON-Dateien). Insgesamt wird für alle Testsets Speicherplatz von ca. 49,2 GB benötigt.

6 Evaluation

Dieses Kapitel erläutert das Vorgehen bei der Evaluation des Algorithmus und stellt die Ergebnisse vor. Kapitel 6.1 betrachtet die verschiedenen Frequenzerkennungsverfahren. Kapitel 6.2 evaluiert die verschiedenen Ansätze zur Schwebungserkennung. Abschließend wird in Kapitel 6.3 das Gesamtsystem betrachtet. Dabei wird jeweils auf die verwendete Methodik und die Ergebnisse für die verschiedenen Testsets eingegangen. Für sämtliche Evaluationen werden zuerst nur die Testsets verwendet, bei denen der Pitch-Shifting-Algorithmus aus der librosa-Audiobibliothek verwendet wurde. Ein Vergleich der Ergebnisse mit dem Algorithmus aus der pyrubberband-Audiobibliothek findet sich im Kapitel 6.4.

6.1 Frequenzerkennung

6.1.1 Methodik

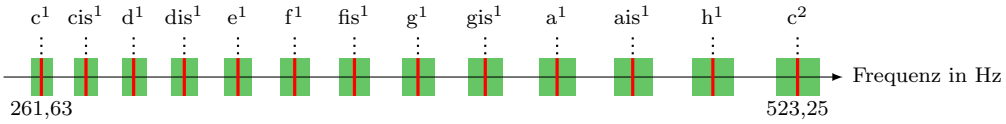
Für die Evaluation der Frequenzerkennung werden nacheinander die verschiedenen Algorithmen getestet. Dabei werden alle Testdaten des verwendeten Testsets für die Ausführung verwendet. Die jeweils 10 Sekunden langen Audiodateien werden in kürzere Abschnitte eingeteilt, wie dies auch beim Gesamtsystem getan wird (siehe Kapitel 4.2). Jedes Fenster hat eine Größe von 8.192 Samples mit einem Abstand von 4.096 Samples zwischen den Startpositionen benachbarter Fenster, so dass sich alle Fenster jeweils zur Hälfte mit dem vorherigen und nächsten Fenster überlappen. Für jede Audiodatei werden insgesamt 100 einzelne Fenster betrachtet. Insgesamt ergibt sich somit für die Gesamtzahl N_{PD} der Testfälle der Frequenzerkennung ein Wert von 696.800.

Auf alle Fenster wird der jeweils zu testenden Frequenzerkennungsalgorithmus angewendet. Die Ergebnisse aller verarbeiteten Fenster können jeweils in eine der drei folgenden Kategorien eingeteilt werden:

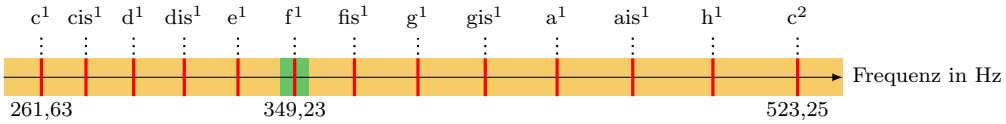
- Korrekt erkannte Frequenzen $C_{\mathrm{PD,K}}$: Dieser Klasse werden alle Ergebnisse zugeordnet, bei denen die Abweichung zwischen der erkannten und erwarteten Frequenz maximal $\pm 25\,\mathrm{Cent}$ beträgt. Diese entsprechen den richtig-positiven Ergebnissen. Die Gesamtzahl der hierin einsortierten Elemente wird als $N_{\mathrm{PD,K}}$ bezeichnet.

- Falsch erkannte Frequenzen $C_{\mathrm{PD,F}}$: Dieser Klasse werden alle Ergebnisse zugeordnet, deren Werte außerhalb der erwarteten Abweichung liegen. Diese entsprechen den falsch-positiven Ergebnissen. Die Gesamtzahl der hierin einsortierten Elemente wird als $N_{\mathrm{PD,F}}$ bezeichnet.
- Nicht erkannte Frequenzen $C_{\mathrm{PD,N}}$: Dieser Klasse werden alle Ergebnisse zugeordnet, bei denen keine Frequenz erkannt wurde. Diese entsprechen den falsch-negativen Ergebnissen. Die Gesamtzahl der hierin einsortierten Elemente wird als $N_{\mathrm{PD,N}}$ bezeichnet.

Die maximale Abweichung von ±25 Cent wurde gewählt, um ausreichend Platz für nicht erkannte Ergebnisse zwischen benachbarten Halbtönen (deren Abstand jeweils 100 Cent beträgt) zu erhalten (siehe Abbildung 6.1). Somit entstehen gleich große Bereiche für korrekt erkannte und falsch erkannte Ergebnisse: jeweils 50 Cent um die Töne herum und 50 Cent zwischen ihnen.



(a) Maximal erlaubte Abweichungen für alle Töne der Oktave. Grün markiert sind jeweils die Bereiche um die Frequenzen, in denen diese als korrekt erkannt klassifiziert werden.



(b) Maximal erlaubte Abweichung für den konkreten Ton f^1. Grün markiert ist der Bereich, in dem die Frequenz als korrekt erkannt klassifiziert wird. In den orangen Bereichen wird die Frequenz hingegen als falsch erkannt klassifiziert.

Abbildung 6.1: Veranschaulichung der maximal erlaubten Abweichung für korrekt erkannte Frequenzen. Die roten Markierungen entsprechenden den Frequenzen der Töne der gleichstufigen Stimmung zwischen c^1 (261,63 Hz) und c^2 (523,25 Hz).

Für den Vergleich der verschiedenen Frequenzerkennungsverfahren wird die Trefferquote R (englisch recall) und die Genauigkeit P (englisch precision) berechnet. Erstere bezeichnet das Verhältnis aller korrekt erkannten Ergebnisse zur Gesamtzahl der erwarteten Ergebnisse:

$$R = \frac{N_{\mathrm{PD,K}}}{N_{\mathrm{PD}}} \quad \text{(vgl. [DD08], S. 138).} \tag{6.1}$$

Letztere bezeichnet das Verhältnis aller korrekt erkannten Ergebnisse zur Gesamtzahl der gefundenen Ergebnisse:

$$P = \frac{N_{\mathrm{PD,K}}}{N_{\mathrm{PD,K}} + N_{\mathrm{PD,F}}} \quad \text{(vgl. [DD08], S. 138).} \tag{6.2}$$

Zusätzlich wird der Mittelwert $\bar{D}$ der Abweichungen aller korrekt erkannten Ergebnisse berechnet.

Für die Detailanalyse werden diese Werte sowohl für die Gesamtmenge der Ergebnisse als auch aufgeschlüsselt nach den erwarteten Frequenzen erfasst. Dazu wird der Frequenzbereich der Testdaten in 17 Klassen eingeteilt. Die Einteilung wird so gewählt, dass jede Klasse exakt den Tonumfang einer halben Oktave (also 6 HTS) umfasst. Die Indizierung der Klassen erfolgt mithilfe der MIDI-Notennummern[1]. Ein Überblick über die konkreten Klassen, inklusive der Frequenzbereiche und Anzahl der Testfälle, ist in Tabelle 6.1 gegeben.

Für jede Implementierung wird die beste Konfiguration mit einem iterativen Vorgehen ermittelt. Dies geschieht unter Zuhilfenahme des reduzierten Testset *Additive 1 reduced* (Nr. 2). Dazu werden die Implementierungen mit verschiedenen Parametrierungen getestet und die Parameter der jeweiligen Verfahren iterativ so lange angepasst, bis jeweils ein Maximum der korrekt erkannten Ergebnisse bzw. der Trefferquote gefunden wird. Diese Konfiguration wird dann verwendet, um die Tests mit den vollständigen Testdaten *Additive 1 complete* (Nr. 1) und *GrandOrgue 1 complete* (Nr. 5) durchzuführen. Die dazugehörigen Ergebnisse werden in den beiden folgenden Abschnitten genauer analysiert.

6.1.2 Ergebnisse für additiven Testdaten

Durch das iterative Vorgehen zum Finden der besten Konfiguration wurden die zu verwendenden Parameter für die einzelnen Verfahren ermittelt. Die exakten Werte der Parameter sind in der Tabelle A.1 im Anhang gegeben. Die resultierenden Trefferquoten und Genauigkeiten sind in Abbildung 6.2 dargestellt. In Abbildung 6.3 finden sich die durchschnittlichen Abweichungen für die korrekt erkannten Frequenzen.

[1] Im MIDI-Standard wird jedem Ton ein Index zwischen 0 und 127 zugewiesen (vgl. [Joa15], S. 250). Die dazugehörigen Frequenzen basieren auf der gleichstufigen Stimmung, bei der der Kammerton a mit der Notennummer 69 einer Frequenz von 440 Hz entspricht (vgl. [Joa15], S. 383 ff.). Die Umrechnung der Notennummern i zu Frequenzen f ist folglich, basierend auf der Formel 2.5, folgendermaßen möglich:

$$f = (\sqrt[12]{2})^{(i-69)} \cdot 440\,\mathrm{Hz}. \tag{6.3}$$

MIDI-Noten-nummern	**Frequenzbereiche**	**Anzahl Testfälle**
[6; 12)	[11,56 Hz; 16,35 Hz)	100
[12; 18)	[16,35 Hz; 23,12 Hz)	3.400
[18; 24)	[23,12 Hz; 32,70 Hz)	5.100
[24; 30)	[32,70 Hz; 46,25 Hz)	33.000
[30; 36)	[46,25 Hz; 65,41 Hz)	38.200
[36; 42)	[65,41 Hz; 92,50 Hz)	75.600
[42; 48)	[92,50 Hz; 130,81 Hz)	74.900
[48; 54)	[130,81 Hz; 185,00 Hz)	91.800
[54; 60)	[185,00 Hz; 261,63 Hz)	76.700
[60; 66)	[261,63 Hz; 369,99 Hz)	77.800
[66; 72)	[369,99 Hz; 523,25 Hz)	62.300
[72; 78)	[523,25 Hz; 739,99 Hz)	61.700
[78; 84)	[739,99 Hz; 1046,50 Hz)	39.000
[84; 90)	[1046,50 Hz; 1479,98 Hz)	31.200
[90; 96)	[1479,98 Hz; 2093,00 Hz)	15.600
[96; 102)	[2093,00 Hz; 2959,96 Hz)	7.800
[102; 108)	[2959,96 Hz; 4186,01 Hz)	2.600

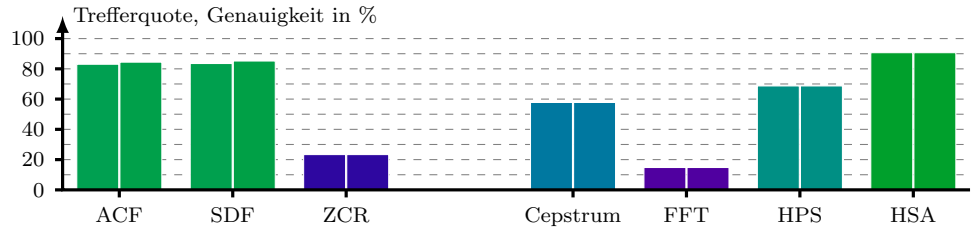
Tabelle 6.1: Einteilung der Frequenzbereiche für die Testfälle der Frequenzerkennung.

Betrachtung von Trefferquote und Genauigkeit

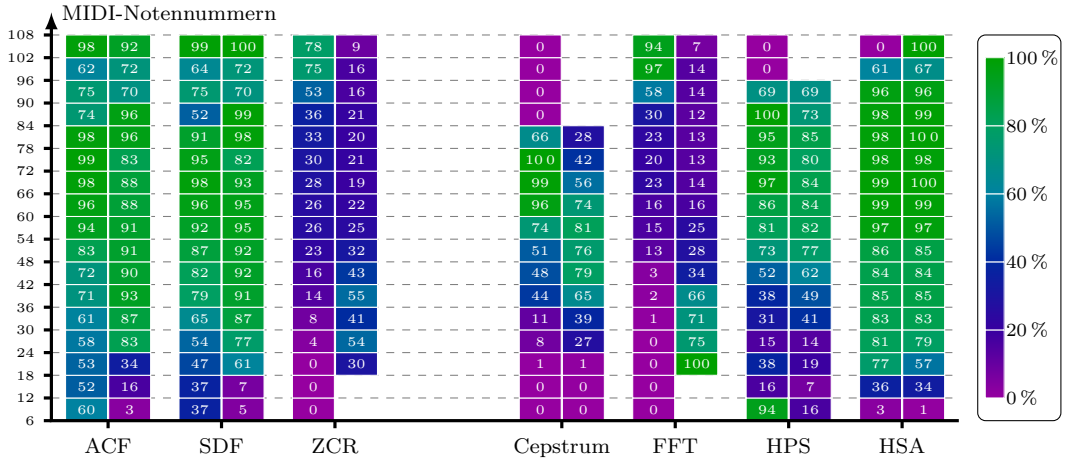
Beim Vergleich der zusammengefassten Trefferquoten und Genauigkeiten zeigt sich, dass zwischen diesen kaum Unterschiede bestehen. Bei ZCR, Cepstrum, FFT, HPS und HSA sind die Werte identisch. Bei ACF und SDF liegen die Differenzen bei 1,38 % bzw. 1,65 %.

Global betrachtet liefert die Frequenzerkennung per HSA mit 90,67 % die besten Ergebnisse. Darauf folgen dann SDF und ACF mit Trefferquoten von 83,52 % bzw. 83,04 %. Mit 68,70 % und 57,77 % liegen mit einigem Abstand das HPS- und Cepstrum-Verfahren. Die schlechtesten Ergebnisse liefern die Frequenzerkennungen per ZCR (23,32 %) und FFT (14,78 %).

Entsprechend sind auch die Ergebnisse auf der Heatmap. Bei ACF, SDF und HSA werden über einen Großteil des Frequenzbereichs hohe Trefferquoten erreicht. Bei HSA werden die höchsten Trefferquoten von ca. 80 % bis 100 % im Bereich der Töne 24 – 96 erreicht (dies entspricht dem Frequenzbereich von 32,70 Hz bis 2093,00 Hz). Bei ACF und SDF ist dieser Bereich maximaler Ergebnisse kleiner und umfasst die Tonnummern 48 –



(a) Zusammengefasste Trefferquoten (links) und Genauigkeiten (rechts) für die kompletten Testsets.



(b) Heatmap mit den Trefferquoten (links) und Genauigkeiten (rechts) für verschiedene Frequenzbereiche. Die darin angegebenen Werte wurden jeweils auf ganze Prozente gerundet.

Abbildung 6.2: Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Frequenzerkennungsverfahren mit additiven Testdaten.

84 bzw. 42 – 84. Dafür sind dort die Trefferquoten in den äußeren Bereichen, also bei den Tönen 6 – 24 und 96 – 108, höher als bei der HSA. Bei diesen drei Verfahren zeigen sich ebenfalls ähnliche Werte für die Genauigkeiten. Bei den anderen Verfahren ZCR, Cepstrum, FFT und HPS sind die Bereiche mit niedrigeren Trefferquoten und Genauigkeiten deutlich stärker ausgeprägt. So finden sich bei ZCR z.B. nur drei Frequenzbereiche mit einer Trefferquote von mehr als 50 %. Ausgehend von der Betrachtung der Trefferquoten und Genauigkeiten liefern also die HSA-, SDF- und ACF-Verfahren die besten Ergebnisse.

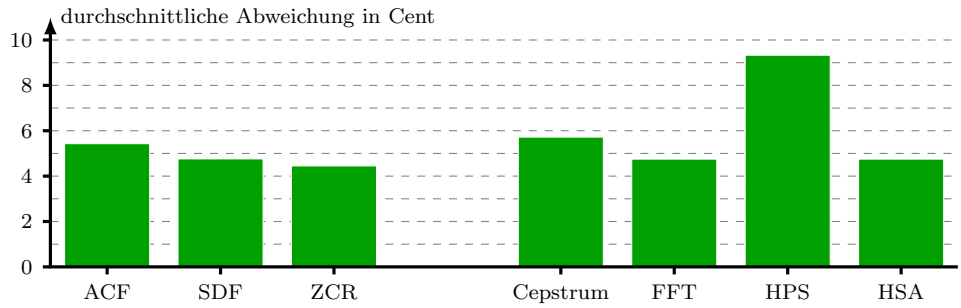


Abbildung 6.3: Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Frequenzen für die verschiedenen Frequenzerkennungsverfahren mit additiven Testdaten.

Betrachtung der durchschnittlichen Abweichungen

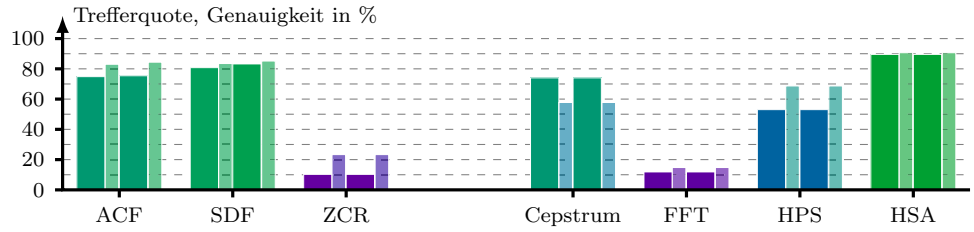
Die niedrigsten durchschnittlichen Abweichungen liegen zwischen 4,45 Cent und 4,76 Cent und wurden bei SDF, ZCR, FFT und HSA erreicht. Etwas erhöhte Werte liegen mit 5,43 % und 5,72 % für ACF und das Cepstrum vor. Lediglich bei der HPS-basierten Frequenzerkennung ist die Abweichung mit 9,33 Cent deutlich höher. Es lässt sich also feststellen, dass die Verfahren mit den höchsten Trefferquoten (HSA, SDF und ACF) auch zu den Verfahren mit den niedrigsten durchschnittlichen Abweichungen gehören.

6.1.3 Ergebnisse für GrandOrgue-basierte Testdaten

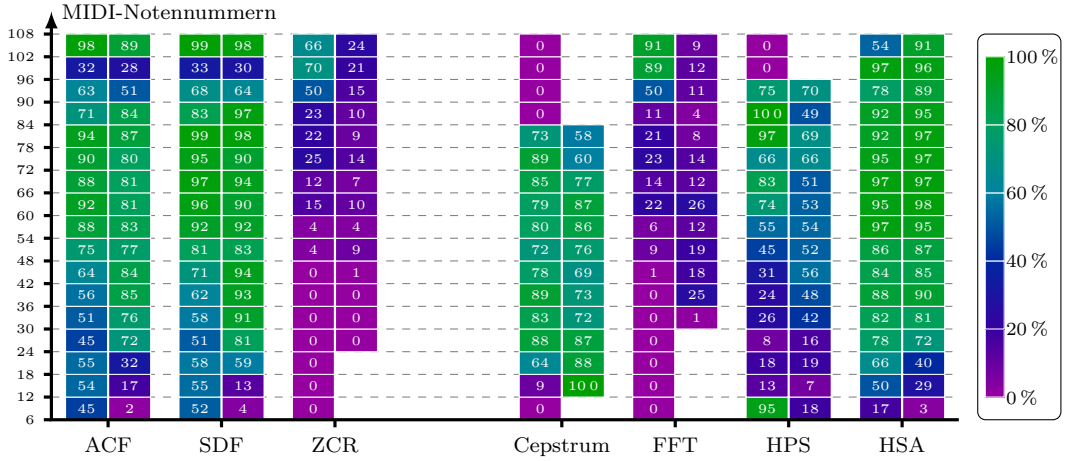
Für die Evaluation mit den *GrandOrgue*-basierten Testdaten werden für die Frequenzerkennungsverfahren dieselben Parametrierungen wie zuvor verwendet. Die exakten Werte der Parameter sind ebenfalls in der Tabelle A.1 im Anhang gegeben. Die resultierenden Trefferquoten und Genauigkeiten sind in Abbildung 6.4 dargestellt. In Abbildung 6.5 finden sich die durchschnittlichen Abweichungen für die korrekt erkannten Frequenzen.

Betrachtung von Trefferquote und Genauigkeit

Der Vergleich der zusammengefassten Ergebnisse zeigt, dass hier ebenfalls kaum Unterschiede zwischen Trefferquoten und Genauigkeiten besteht. Auch die Frequenzerkennung per HSA liefert mit 89,56 % die besten Ergebnisse. Darauf folgen SDF mit 80,81 % und ACF mit 74,86 %. Im Gegensatz zu vorher hat das Cepstrum eine ähnlich hohe Trefferquote (74,04 %). Es ist das einzige Verfahren, bei dem die Ergebnisse mit den GrandOrgue-basierten Testdaten besser sind als mit den additiven Daten. Die Zunahme



(a) Zusammengefasste Trefferquoten (links) und Genauigkeiten (rechts) für die kompletten Testsets. Zum Vergleich sind die Ergebnisse für die additiven Testdaten aus Abbildung 6.2 als dünnere Balken ebenfalls dargestellt.



(b) Heatmap mit den Trefferquoten (links) und Genauigkeiten (rechts) für verschiedene Frequenzbereiche. Die darin angegebenen Werte wurden jeweils auf ganze Prozente gerundet.

Abbildung 6.4: Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Frequenzerkennungsverfahren mit GrandOrgue-basierten Testdaten.

beträgt dabei 16,27 %. Ein Grund dafür könnte sein, dass die Testdaten mehr Obertöne enthalten und sich somit die Periodizität im Amplitudenspektrum besser erkennen lässt.

Bei allen anderen Verfahren ist die Trefferquote schlechter als mit den additiven Daten. Beim HSA-Verfahren ist der Unterschied mit 1,11 % am geringsten. Bei SDF und FFT sind die Differenzen mit 2,71 % bzw. 2,90 % ebenfalls relativ gering. Für die restlichen Verfahren sind die Unterschiede teilweise deutlich größer (ACF: 8,18 %, ZCR: 13,07 %, HPS: 15,65 %).

Die Heatmap der Ergebnisse für die GrandOrgue-basierten Testdaten sieht ähnlich aus wie bei den additiven Testdaten. Bei HSA, SDF und ACF überwiegen die Anteile mit hohen Trefferquoten. Ein deutlicher Unterschied ist, dass bei dem Cepstrum der Bereich

mit hohen Trefferquoten größer ist. Damit zeigt sich die erhöhte Trefferquote ebenfalls in der Heatmap. Bei den anderen Verfahren (ZCR, FFT und HPS) hingegen überwiegen trotzdem weiterhin die Bereiche mit schlechten Ergebnissen. Ein Grund für die deutlich schlechteren Ergebnisse der ZCR könnten ebenfalls die komplexeren Wellenformen der Testdaten sein. Diese führen zu mehr gezählten Nulldurchläufen und somit zu falschen Ergebnissen.

Betrachtung der durchschnittlichen Abweichungen

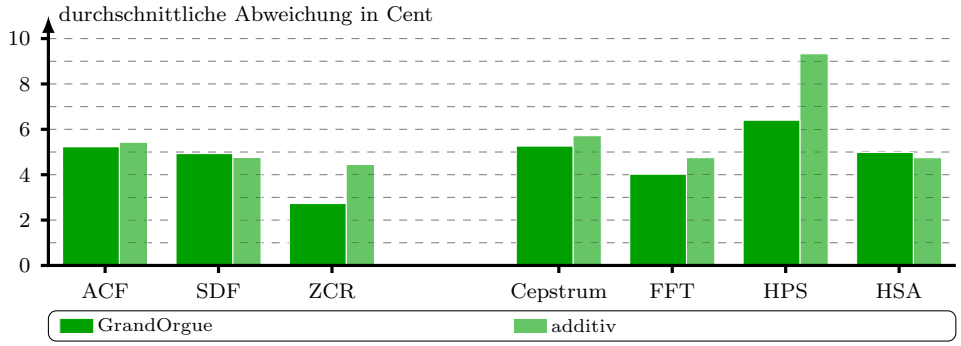


Abbildung 6.5: Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Frequenzen für die verschiedenen Frequenzerkennungsverfahren mit GrandOrgue-basierten Testdaten. Zum Vergleich sind die Ergebnisse für die additiven Testdaten aus Abbildung 6.3 als dünnere Balken ebenfalls dargestellt.

Die durchschnittlichen Abweichungen zeigen ähnliche Werte im Vergleich zu den Tests mit den additiven Daten. Für fast alle Verfahren liegen die Abweichungen prinzipiell in einem ähnlichen Bereich, teilweise kleiner (ACF, Cepstrum, FFT) oder größer (SDF, HSA). Lediglich bei ZCR und HPS sind die Unterschiede deutlicher zu sehen. In beiden Fällen sind die durchschnittlichen Abweichungen bei den GrandOrgue-basierten Testdaten geringer als bei Verwendung der additiven Testdaten (um 1,72 Cent bzw. 2,93 Cent). Die Ursache dafür könnte in den geringeren Trefferquoten für diese Verfahren liegen. Indem Testfälle, die bei den additiven Testdaten zu hohen Abweichungen geführt haben, nicht mehr erkannt werden und damit herausfallen, kann die durchschnittliche Abweichung sinken.

Die Reihenfolge zwischen den verschiedenen Verfahren ändert sich dadurch aber nicht. Die ZCR hat weiterhin die niedrigste und das HPS die höchste durchschnittliche Abweichung.

Fazit

Im Vergleich zu den obigen Ergebnissen kann festgestellt werden, dass die HSA- und SDF-Verfahren weiterhin am besten für die Frequenzerkennung geeignet ist. Der Unterschied der Trefferquoten zwischen ACF und SDF ist hingegen von 0,48 % auf 5,95 % deutlich gestiegen. Somit stellt das ACF-Verfahren keine Alternative zur SDF mehr dar. Aufgrund der höheren Trefferquote bei ähnlicher Abweichung, wird für die Tests des Gesamtsystems die HSA-basierte Frequenzerkennung verwendet.

6.2 Schwebungserkennung

6.2.1 Methodik

Für die Evaluation der Schwebungserkennung wird ähnlich wie bei der Frequenzerkennung vorgegangen. Dazu werden alle Testdaten des verwendeten Testsets in Fenster zu je 4 Sekunden Länge (176.400 Samples) unterteilt. Dies entspricht exakt der Fenstergröße, die auch für das Gesamtsystem verwendet wird (siehe Kapitel 4.1). Die Schrittweite zwischen zwei benachbarten Fenstern beträgt jeweils 0,25 Sekunden bzw. 11.025 Samples. Die Überlappung beträgt somit 93,75 %. Für jede Audiodatei werden insgesamt 22 einzelne Fenster betrachtet. Insgesamt ergibt sich somit für die Gesamtzahl N_{BD} der Testfälle der Schwebungserkennung ein Wert von 153.296.

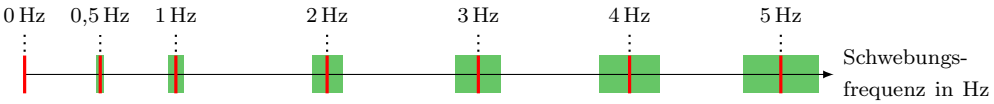
Auf alle Fenster wird der jeweils zu testende Schwebungserkennungsalgorithmus angewendet. Die Ergebnisse aller verarbeiteten Fenster werden jeweils in eine der drei folgenden Kategorien eingeteilt:

- Korrekt erkannte Schwebungen $C_{\mathrm{BD,K}}$: Dieser Klasse werden alle Ergebnisse zugeordnet, bei denen die Abweichung zwischen der erkannten und erwarteten Schwebungsfrequenz maximal $\pm 5\,\%$ beträgt. Diese entsprechen den richtig-positiven Ergebnissen. Die Gesamtzahl der hierin einsortierten Elemente wird als $N_{\mathrm{BD,K}}$ bezeichnet.

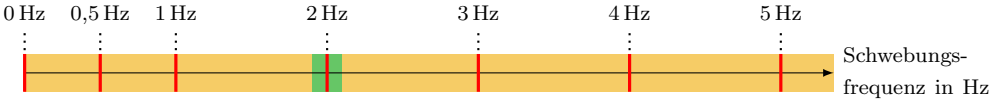
- Falsch erkannte Schwebungen $C_{\mathrm{BD,F}}$: Dieser Klasse werden alle Ergebnisse zugeordnet, deren Werte außerhalb der erwarteten Abweichung liegen. Diese entsprechen den falsch-positiven Ergebnissen. Die Gesamtzahl der hierin einsortierten Elemente wird als $N_{\mathrm{BD,F}}$ bezeichnet.

- Nicht erkannte Schwebungen $C_{\mathrm{BD,N}}$: Dieser Klasse werden alle Ergebnisse zugeordnet, bei denen keine Schwebungsfrequenz erkannt wurde. Diese entsprechen den falsch-negativen Ergebnissen. Die Gesamtzahl der hierin einsortierten Elemente wird als $N_{\mathrm{BD,N}}$ bezeichnet.

Die Verwendung einer maximal erlaubten relativen Abweichung hat zur Folge, dass bei höheren Schwebungsfrequenzen auch höhere Abweichungen möglich sind (siehe Abbildung 6.6). So werden Ergebnisse, bei denen keine Schwebung erwartet wird (die Schwebungsfrequenz beträgt 0 Hz), nur als korrektes Ergebnis gewertet, wenn auch exakt dieses Ergebnis vorliegt. Bei einer Schwebungsfrequenz von 1 Hz hingegen werden Ergebnisse zwischen 0,95 Hz und 1,05 Hz als korrekt erkannt gewertet. Somit wird für Ergebnisse mit niedrigeren Schwebungsfrequenzen implizit eine höhere absolute Genauigkeit vorausgesetzt als bei höheren.



(a) Maximal erlaubte Abweichungen für die verschiedenen Schwebungen. Grün markiert sind jeweils die Bereiche um die Schwebungsfrequenzen, in denen diese als korrekt erkannt klassifiziert werden.



(b) Maximal erlaubte Abweichung für die konkrete Schwebungsfrequenz von 2 Hz. Grün markiert ist der Bereich, in dem die Schwebungsfrequenz als korrekt erkannt klassifiziert wird. In den orangen Bereichen wird die Schwebungsfrequenz hingegen als falsch erkannt klassifiziert.

Abbildung 6.6: Veranschaulichung der maximal erlaubten Abweichung für korrekt erkannte Schwebungsfrequenzen. Die roten Markierungen entsprechen verschiedenen Schwebungsfrequenzen zwischen 0 Hz und 5 Hz.

Für den Vergleich der verschiedenen Schwebungserkennungsverfahren wird ebenfalls die Trefferquote R, die Genauigkeit P und der Mittelwert $\bar{D}$ der Abweichungen aller korrekt erkannten Ergebnisse berechnet. Dies erfolgt analog wie bei der Frequenzerkennung:

$$R = \frac{N_{\mathrm{BD,K}}}{N_{\mathrm{BD}}}, \tag{6.4}$$

$$P = \frac{N_{\mathrm{BD,K}}}{N_{\mathrm{BD,K}} + N_{\mathrm{BD,F}}}. \tag{6.5}$$

Für die Detailanalyse werden die Ergebnisse ebenfalls für die Gesamtmenge der Testfälle als auch für einzelne Klassen, basierend auf der erwarteten Schwebungsfrequenz, berechnet. Insgesamt kommen dabei 9 Klassen zum Einsatz, deren Grenzen detailliert in Tabelle 6.2 gegeben sind. Diese sind so angeordnet, dass jede Klasse möglichst exakt

doppelte so viele Frequenzen wie die vorherige Klasse enthält. Außer bei den ersten drei Klassen ist dies immer gegeben. Die erste Klasse enthält nur Testfälle, bei denen keine Schwebung vorliegt, die Schwebungsfrequenz also exakt 0 Hz beträgt. Die zweite und dritte Klasse teilt den Frequenzbereich bis 0,70 Hz exakt in der Mitte bei 0,35 Hz. Durch diese Einteilung wird eine annähernd regelmäßige Verteilung der Testfälle auf die Klassen erreicht. Extreme Abweichungen davon zeigen sich nur in den höchsten Klassen.

Frequenzbereiche	**Anzahl Testfälle**
[0 Hz; 0 Hz]	12.584
(0 Hz; 0,35 Hz]	15.136
(0,35 Hz; 0,70 Hz]	21.384
(0,70 Hz; 1,41 Hz]	25.168
(1,41 Hz; 2,82 Hz]	25.168
(2,82 Hz; 5,65 Hz]	24.992
(5,65 Hz; 11,31 Hz]	19.800
(11,31 Hz; 22,62 Hz]	8.008
(22,62 Hz; 45,25 Hz]	1.056

Tabelle 6.2: Einteilung der Frequenzbereiche für die Testfälle der Schwebungserkennung.

Für jede Implementierung wird die beste Konfiguration in einem iterativen Prozess bestimmt. Dafür wird ebenfalls das reduzierte Testset *Additive 1 reduced* (Nr. 2) verwendet. Der Wert des Parameters V_{nobeat} wird für die Evaluation mit den additiven Testdaten zuerst auf 0,95 festgelegt. Somit wird davon ausgegangen, dass keine Schwebung vorliegt, wenn das Verhältnis vom globalen Minimum zum globalen Maximum der betrachteten Daten größer als 95 % ist. Die resultierenden Konfigurationen wurden für die Evaluation mit den vollständigen additiven Testdaten (*Additive 1 complete* (Nr. 1)) verwendet.

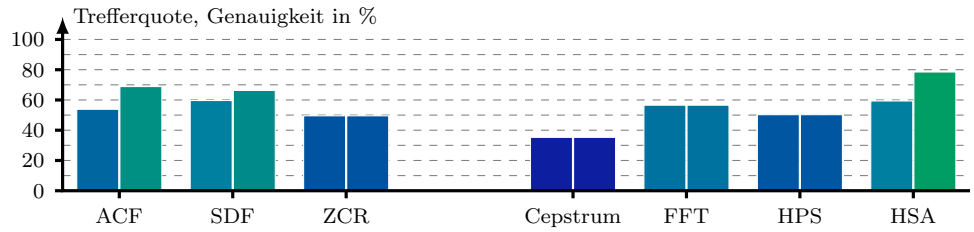
Für die folgende Evaluation mit den GrandOrgue-basierten Testdaten wurden alle Parameter außer V_{nobeat} unverändert gelassen und nur dieser angepasst. Dafür kam das reduzierte Testset *GrandOrgue 1 reduced* (Nr. 6) zum Einsatz. Nachdem jeweils die Konfigurationen mit den höchsten Trefferquoten gefunden wurden, wurde das Testset *GrandOrgue 1 complete* (Nr. 5) zur Ermittlung der resultierenden Trefferquoten und Genauigkeiten verwendet.

6.2.2 Ergebnisse für additive Testdaten

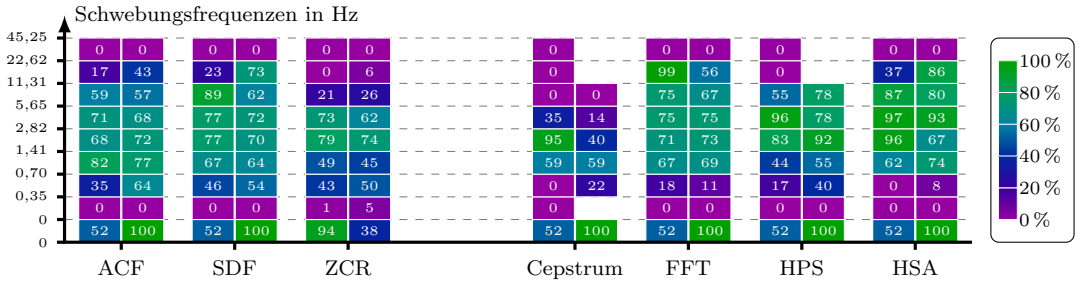
Durch das iterative Vorgehen zum Finden der besten Konfiguration wurden die verwendeten Parameter für die einzelnen Verfahren ermittelt. Die exakten Werte der Parameter

sind in den Tabellen A.2, A.3 und A.4 im Anhang gegeben. Die resultierenden Trefferquoten und Genauigkeiten sind in den Abbildungen 6.7, 6.8 und 6.9 dargestellt. In Abbildung 6.10 finden sich die durchschnittlichen Abweichungen für die korrekt erkannten Schwebungen.

Ergebnisse bei Analyse der Amplituden



(a) Zusammengefasste Trefferquoten (links) und Genauigkeiten (rechts) für die kompletten Testsets.



(b) Heatmap mit den Trefferquoten (links) und Genauigkeiten (rechts) für verschiedene Schwebungsfrequenzbereiche. Die darin angegebenen Werte wurden jeweils auf ganze Prozente gerundet.

Abbildung 6.7: Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Amplitude mit additiven Testdaten.

Bei den Ergebnissen für die Analyse der Amplituden (siehe Abbildung 6.7) werden Trefferquoten von knapp 60,00 % erreicht (SDF: 59,62 %; HSA: 59,33 %). Darauf folgen die Analyse per FFT (56,52 %), ACF (53,84 %), HPS (50,26 %) und ZCR (49,51 %). Die schlechtesten Ergebnisse liefert die Analyse per Cepstrum mit 35,28 %.

Bei ZCR, Cepstrum, FFT und HPS stimmen die Genauigkeiten und Trefferquoten exakt überein. Bei den anderen Verfahren sind die Genauigkeiten hingegen höher. Dort ist die Wahrscheinlichkeit, dass ein gefundenes Ergebnis korrekt ist, also höher. Die höchste

Genauigkeit wird bei Verwendung der HSA mit 78,48 % erreicht. Darauf folgen die ACF mit 68,85 % und SDF mit 66,29 %.

Mit diesen Ergebnissen lässt sich festhalten, dass für die Schwebungserkennung durch Analyse der Amplituden am besten die Verwendung von SDF und HSA geeignet ist. Dort ergeben sind jeweils die höchsten Trefferquoten und Genauigkeiten.

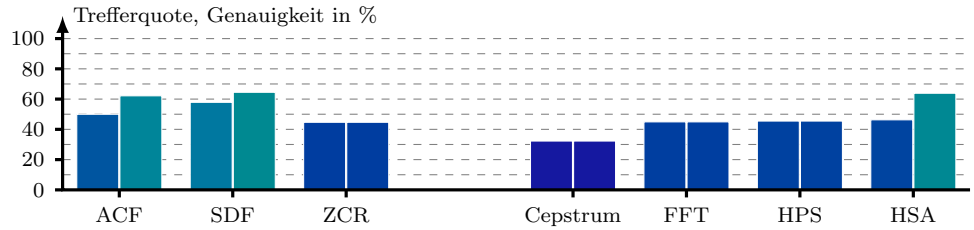
Beim Blick auf die dazugehörige Heatmap zeigt sich, dass alle Verfahren Probleme haben niedrige und hohe Schwebungsfrequenzen (bis 0,70 Hz und ab 11,31 Hz) zu erkennen. Die Erkennung von nicht vorhandenen Schwebungen ist nur zuverlässig mit einer Trefferquote von 52,11 % möglich. Bei der Verwendung der ZCR wird zwar eine Trefferquote von 93,56 % erreicht. Die dazugehörige Genauigkeit ist mit 38,32 % allerdings so gering, dass die Ergebnisse nicht als zuverlässig bezeichnet werden können. Im Gegensatz dazu ist die Genauigkeit bei den anderen Verfahren mit 100 % maximal. Im Bereich zwischen 0 Hz und 0,70 Hz erreicht keines der Verfahren eine Trefferquote von mehr als 50 %. Auch für die hohen Frequenzen werden, außer bei der FFT, keine Trefferquoten von mehr als 37 % erreicht. Für die höchste Klasse liegt die Trefferquote sogar immer bei 0 %.

Ergebnisse bei Analyse der Hüllkurve

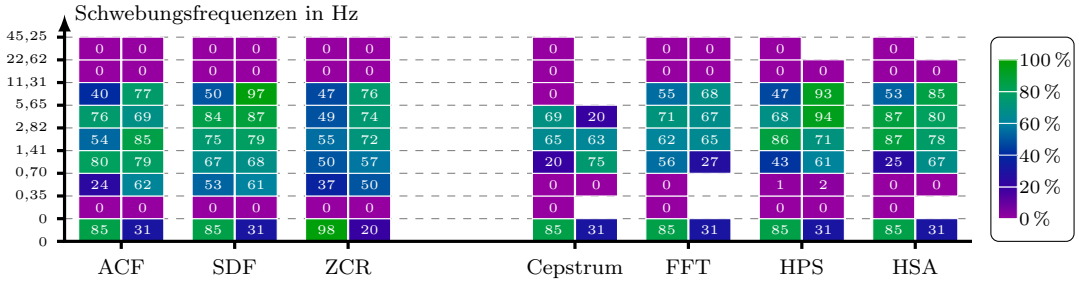
Bei den Ergebnissen für die Analyse der Hüllkurven (siehe Abbildung 6.8) werden geringere Trefferquoten als bei der Analyse der Amplituden erreicht. Die Verschlechterung betragen zwischen 1,75 % bei der SDF und 13,04 % bei der HSA. Auch die Genauigkeiten haben entsprechend geringere Werte. Somit ist die Verwendung dieses Ansatzes, alleine aufgrund der schlechteren Ergebnisse gegenüber der Analyse der Amplituden, nicht empfehlenswert.

Den Grund für die schlechteren Ergebnisse lässt sich in der Heatmap ablesen. Dort sind die Bereiche mit schlechten Trefferquoten deutlich stärker ausgeprägt als bei dem vorher untersuchten Ansatz. So werden Schwebungsfrequenzen über 11,31 Hz nie erkannt. Gleiches gilt bei ACF, SDF und ZCR für die Frequenzen zwischen 0 Hz und 0,35 Hz, bei Cepstrum, FFT, HPS und HSA sogar für die Frequenzen zwischen 0 Hz und 0,70 Hz.

Ein anscheinender Vorteil dieses Ansatzes zeigt sich bei der Erkennung von nicht-vorhandenen Schwebungen. Dort werden Trefferquoten von mindestens 84,68 % erreicht. Die dazugehörigen Genauigkeiten zwischen 20,07 % und 30,56 % zeigen jedoch, dass diese Erkennung nicht zuverlässig funktioniert. Dies spricht ebenfalls gegen eine Verwendung dieses Ansatzes.



(a) Zusammengefasste Trefferquoten (links) und Genauigkeiten (rechts) für die kompletten Testsets.



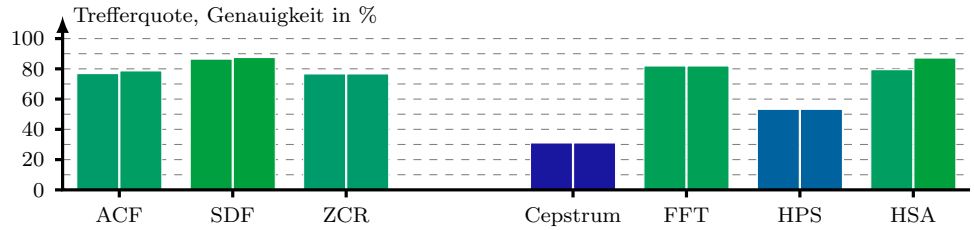
(b) Heatmap mit den Trefferquoten (links) und Genauigkeiten (rechts) für verschiedene Schwebungsfrequenzbereiche. Die darin angegebenen Werte wurden jeweils auf ganze Prozente gerundet.

Abbildung 6.8: Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Hüllkurve mit additiven Testdaten.

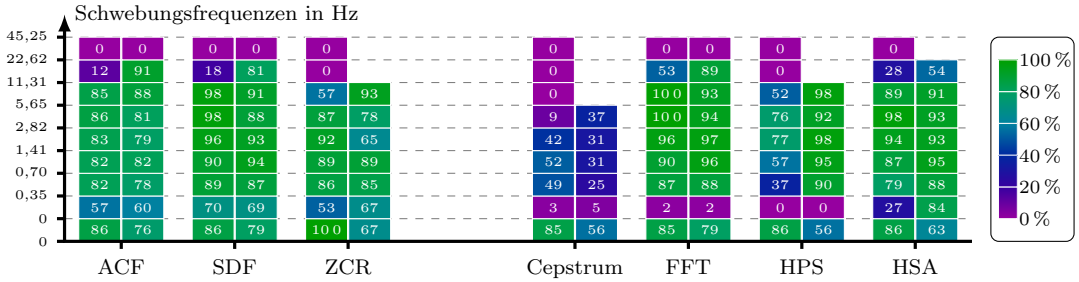
Ergebnisse bei Analyse des Verlaufs der Fourierkoeffizienten

Die Ergebnisse für die Analyse der Fourierkoeffizienten (siehe Abbildung 6.9) zeigen deutlich höhere Trefferquoten und Genauigkeiten als bei den beiden zuvor betrachteten Ansätzen. Bei ACF, SDF, ZCR, FFT und HSA werden Trefferquoten zwischen 76,63 % und 86,42 % erreicht. Die Werte der Genauigkeiten bewegen sich dabei ebenfalls zwischen 76,63 % und 86,42 %. Die Ergebnisse für das Cepstrum und HPS sind hingegen deutlich schlechter. Die besten Ergebnisse liefert die Analyse per SDF, gefolgt von FFT und HSA.

Die dazugehörige Heatmap zeigt, dass die Trefferquoten bei fast allen Schwebungsfrequenzen höher sind als bei der Analyse der Amplituden. Für die Fälle ohne Schwebung (Trefferquote von mehr als 84 %) als auch bei den Schwebungsfrequenzen zwischen 0 Hz und 0,70 Hz fällt dies besonders stark auf. Trotzdem zeigen sich für die Schwebungen bis 0,35 Hz schlechtere Ergebnisse als bei den umliegenden Klassen. Ein Grund dafür könnte sein, dass diese Klasse die Schwebungsfrequenzen unterhalb von 0,25 Hz enthält. In 4.1



(a) Zusammengefasste Trefferquoten (links) und Genauigkeiten (rechts) für die kompletten Testsets.



(b) Heatmap mit den Trefferquoten (links) und Genauigkeiten (rechts) für verschiedene Schwebungsfrequenzbereiche. Die darin angegebenen Werte wurden jeweils auf ganze Prozente gerundet.

Abbildung 6.9: Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit additiven Testdaten.

wurde dieser Wert als niedrigste zu erkennende Schwebungsfrequenz angenommen (bei der Fenstergröße von 4 Sekunden wäre dann exakt eine vollständige Schwebung im Fenster enthalten). Die Testdaten enthalten allerdings mit 0,1 Hz und 0,2 Hz auch Schwebungen geringerer Frequenzen. Diese werden entsprechend nicht korrekt erkannt und führen zu diesem Knick bei den Trefferquoten. Kaum ein Unterschied zeigt sich hingegen bei den Schwebungsfrequenzen über 11,31 Hz. Dort sind die Trefferquoten ähnlich schlecht, die Genauigkeiten allerdings höher.

Hier zeigt sich bereits eine Überlegenheit dieses Ansatzes gegenüber den anderen beiden Verfahren. Die Trefferquoten und Genauigkeiten sind gegenüber der Analyse der Amplituden um bis zu 27,12 % höher.

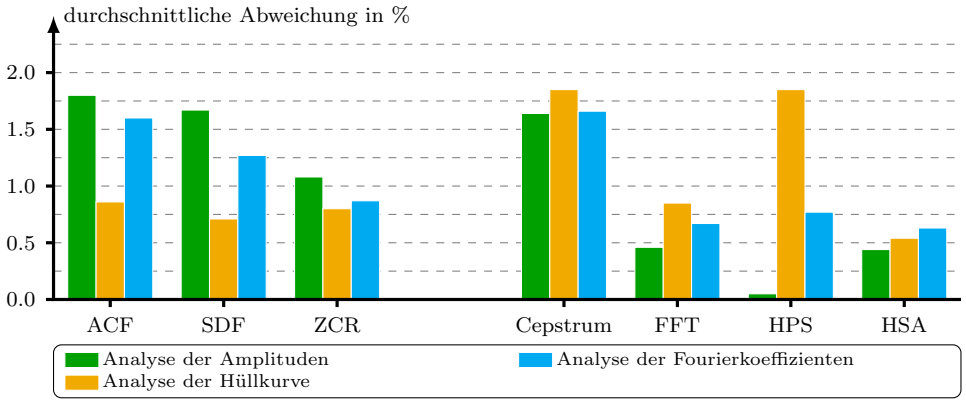


Abbildung 6.10: Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen für die verschiedenen Verfahren zur Schwebungserkennung mit additiven Testdaten.

Betrachtung der durchschnittlichen Abweichungen

Der Vergleich der durchschnittlichen Abweichungen ist in Abbildung 6.10 gegeben. Dort zeigen sich deutliche Unterschiede zwischen den verschiedenen Ansätzen und Frequenzerkennungsverfahren. Die Spanne der Werte reicht von 0,05 % bis 1,85 %. Bei den Frequenzerkennungsverfahren im Zeitbereich (ACF, SDF und ZCR) sind die Abweichungen bei der Analyse der Amplituden jeweils höher als bei der Analyse der Fourierkoeffizienten. Im Frequenzbereich hingegen zeigt sich ein entgegengesetztes Bild. Bei FFT, HPS und HSA sind die Ergebnisse bei Analyse der Amplituden genauer als bei Analyse der Fourierkoeffizienten. Für die Analyse der Fourierkoeffizienten ist die höchste Abweichung im Frequenzbereich (Analyse mit HPS, das Cepstrum wird hier aufgrund der schlechten Trefferquoten nicht weiter berücksichtigt) allerdings immer noch geringer als die kleinste Abweichung im Zeitbereich (Analyse der Fourierkoeffizienten mit ZCR).

Fazit

Für die additiven Testdaten lässt sich festhalten, dass die höchsten Trefferquoten und Genauigkeiten bei der Analyse der Fourierkoeffizienten erreicht werden können. Am besten eignen sich dabei die Analyse per SDF, FFT und HSA. Dabei hat die Verwendung der SDF eine höhere Trefferquote sowie eine höhere durchschnittliche Abweichung. Die anderen Ansätze (FFT und HSA) hingegen haben eine geringere Trefferquote und dafür eine nur etwa halb so große durchschnittliche Abweichung.

6.2.3 Ergebnisse für GrandOrgue-basierte Testdaten

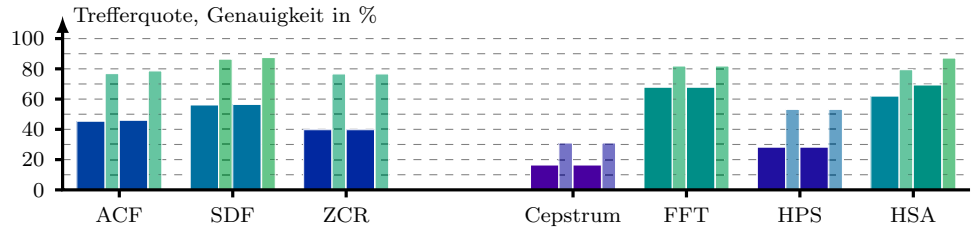
Für die Evaluation mit den *GrandOrgue*-basierten Testdaten werden für die Schwebungserkennung die Parametrierungen von zuvor angepasst. Dazu wird in einem iterativen Vorgehen ein optimaler Wert für den Parameter V_{nobeat} gesucht. Die resultierenden Konfigurationen wurde für die Ausführung der Testsets verwendet. Die exakten Werte der Parameter sind in der Tabelle A.5 im Anhang gegeben. Da die Ergebnisse der Evaluation mit den additiven Daten gezeigt haben, dass die Analyse der Verläufe der Fourierkoeffizienten mit deutlichem Abstand die besten Ergebnisse liefert, wird nur dieser Ansatz weiter betrachtet und auf die Analyse der Amplituden und der Hüllkurve verzichtet. Die Trefferquoten und Genauigkeiten sind in der Abbildung 6.11 dargestellt. In Abbildung 6.12 finden sich die durchschnittlichen Abweichungen für die korrekt erkannten Schwebungen.

Ergebnisse bei Analyse des Verlaufs der Fourierkoeffizienten

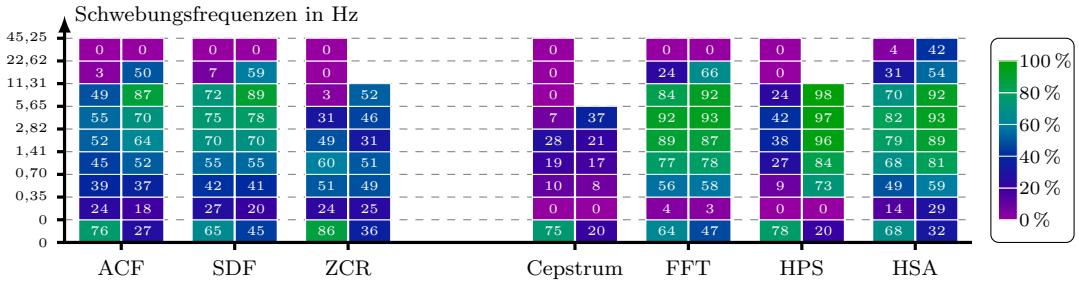
Allgemein lässt sich feststellen, dass die Ergebnisse für die GrandOrgue-basierten Testdaten schlechter sind als für die additiven Testdaten. In allen Fällen sind die Trefferquoten und Genauigkeiten geringer. Die höchsten Trefferquoten liefern weiterhin die Analyse per SDF, FFT und HSA. Allerdings hat sich deren Reihenfolge geändert. Die höchste Trefferquote liefert hier die FFT (67,82 %), darauf folgen HSA (61,98 %) und SDF (56,13 %). Bei letzterem ist der Unterschied zum Ergebnis der additiven Daten mit 30,29 % am höchsten. Bei FFT und HSA sind die Abstände mit 14,05 % und 17,46 % geringer. Alle anderen Ansätze haben Trefferquoten von weniger als 50 %.

Der Grund für den großen Unterschied bei der SDF zeigt sich in der Heatmap. Die Trefferquote für die Schwebungsfrequenzen zwischen 0 Hz und 0,35 Hz hat dort von 69,58 % auf 26,72 % deutlich abgenommen. Bei FFT und HSA ist der Unterschied nicht so extrem, da dort die Trefferquoten bei den additiven Testdaten mit 2,16 % und bzw. 27,13 % zuvor nicht so hoch waren. Abgesehen davon zeigen fast alle Klassen schlechtere Ergebnisse als bei den additiven Testdaten. Der Unterschied beträgt dabei bis zu 30 %. Einzige Ausnahme bildet die Klasse mit den höchsten Schwebungsfrequenzen ab 22,62 Hz. Dies liegt vor allem daran, dass dort die Trefferquoten und Genauigkeiten bereits mit den additiven Testdaten relativ gering waren und somit keine so starken Verschlechterungen möglich sind.

Da diese Verschlechterung hier über den gesamten Frequenzverlauf und bei allen Verfahren beobachtet werden können, kann davon ausgegangen werden, dass die zugrundeliegende Extraktion der Fourierkoeffizienten bei den GrandOrgue-basierten Testdaten nicht mehr so zuverlässig wie bei den additiven Testdaten funktioniert.



(a) Zusammengefasste Trefferquoten (links) und Genauigkeiten (rechts) für die kompletten Testsets. Zum Vergleich sind die Ergebnisse für die additiven Testdaten aus Abbildung 6.9 als dünnere Balken ebenfalls dargestellt.



(b) Heatmap mit den Trefferquoten (links) und Genauigkeiten (rechts) für verschiedene Schwebungsfrequenzbereiche. Die darin angegebenen Werte wurden jeweils auf ganze Prozente gerundet.

Abbildung 6.11: Vergleich der Trefferquoten und Genauigkeiten für die verschiedenen Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit GrandOrgue-basierten Testdaten.

Betrachtung der durchschnittlichen Abweichungen

Beim Vergleich der durchschnittlichen Abweichungen zeigt sich ein ähnliches Bild wie bei den Trefferquoten und Genauigkeiten. In fast allen Fällen ist diese höher als bei den Tests mit den additiven Testdaten. Ausnahmen hiervon sind die Analyse per Cepstrum und HPS. Aufgrund deren schlechter Trefferquoten mit 16,44 % bzw. 28,21 % sind diese Verfahren jedoch nicht in der engeren Auswahl für das Gesamtsystem. Von den Ergebnissen mit den höchsten Trefferquoten hat die Analyse per SDF mit 1,79 % weiterhin die höchste durchschnittliche Abweichung. Das entspricht einer Erhöhung von 0,52 % gegenüber den additiven Testdaten. Bei FFT und HSA sind die durchschnittlichen Abweichungen mit ähnlichen Werten von 1,03 % bzw. 0,96 % deutlich geringer. Dort ist auch der Unterschied gegenüber den additiven Testdaten kleiner (0,36 % und 0,33 %).

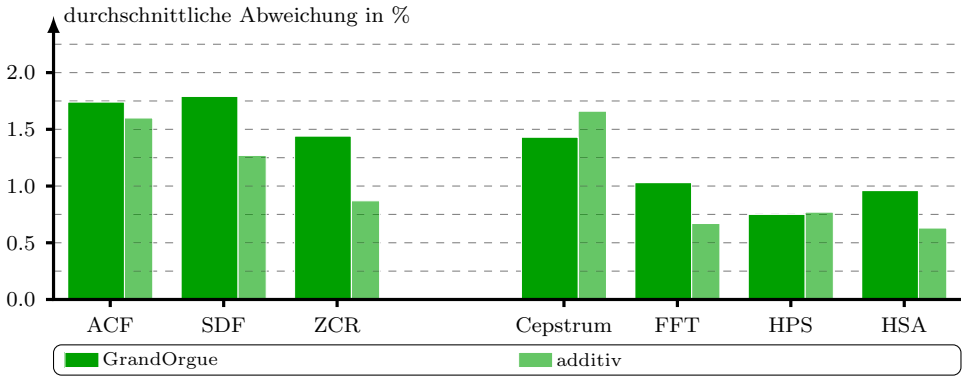


Abbildung 6.12: Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen für die verschiedenen Verfahren zur Schwebungserkennung mit GrandOrgue-basierten Testdaten. Zum Vergleich sind die Ergebnisse für die additiven Testdaten aus Abbildung 6.10 als dünnere Balken ebenfalls dargestellt.

Fazit

Für die GrandOrgue-basierten Testdaten lässt sich feststellen, dass die Analyse der Fourierkoeffizienten per FFT die besten Ergebnisse liefert. Damit konnte die höchste Trefferquote bei vergleichsweise geringer durchschnittlicher Abweichung erreicht werden. Mit einer ähnlich hohen Trefferquote und durchschnittlichen Abweichung folgt darauf die Analyse der Fourierkoeffizienten per HSA. Die drittbesten Ergebnisse liefert, mit etwas Abstand und höheren Abweichungen, die Analyse per SDF.

6.3 Gesamtsystem

6.3.1 Methodik

Für die Evaluation des Gesamtsystems werden die beiden vorherigen Vorgehen, zur Evaluation der Frequenzerkennung (siehe Kapitel 6.1) und der Schwebungserkennung (siehe Kapitel 6.2), kombiniert. Dafür werden dieselben Fenstergrößen und Schrittweiten wie bei der Evaluation der Schwebungserkennung verwendet (Fenstergröße: 4 Sekunden/176.400 Samples, Schrittweite: 0,25 Sekunden/11.025 Samples). Die Klassifizierungen der Ergebnisse in die drei Klassen (korrekt erkannt, falsch erkannt und nicht erkannt) werden ebenfalls übernommen.

Bei der Evaluation des Gesamtsystems wird allerdings keine iterative Suche nach der besten Parametrierung durchgeführt. Stattdessen werden lediglich die Trefferquoten und Genauigkeiten für das Zusammenspiel der besten Frequenz- und Schwebungserkennungsverfahren ermittelt. Für die Frequenzerkennung kommt aufgrund der maximalen Trefferquote das HSA-Verfahren zum Einsatz. Für die Schwebungserkennung wird eine Analyse der Fourierkoeffizienten durchgeführt. Aufgrund ähnlich hoher Trefferquoten und Genauigkeiten werden dafür die drei besten Verfahren untersucht. Konkret sind dies SDF, FFT und HSA. Die restlichen Verfahren (ACF, ZCR, Cepstrum und HPS) werden nicht weiter betrachtet.

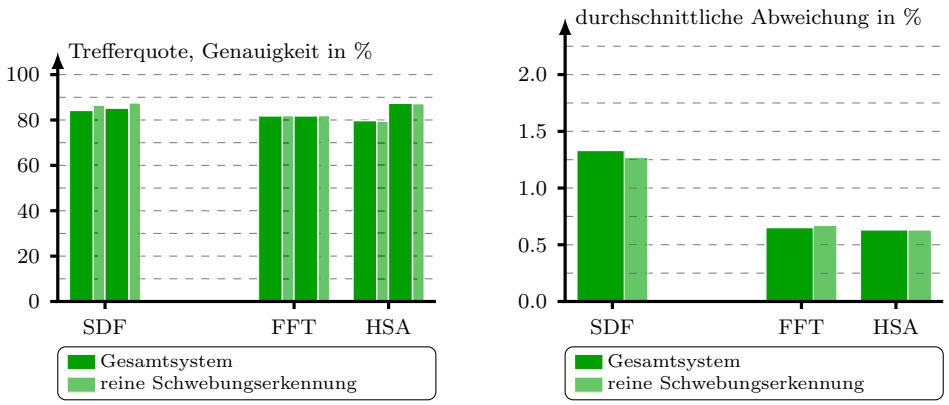
Die Evaluation erfolgt wieder sowohl für die additiven als auch die GrandOrgue-basierten Testdaten. Dafür werden die Testsets *Additive 1 complete* (Nr. 1) und *GrandOrgue 1 complete* (Nr. 5) verwendet.

6.3.2 Ergebnisse für additive Testdaten

Die Trefferquote der Frequenzerkennung im Gesamtsystem für die additiven Testdaten beträgt 91,95 %. Damit liegt diese sogar höher als das Ergebnis der reinen Frequenzerkennung mit 90,67 %. Im Gesamtsystem werden vor jeder Schwebungserkennung 42 Frequenzerkennungen ausgeführt und deren Ergebnisse zu einem zusammengefasst (siehe Kapitel 4.1). Dadurch können einzelne falsche Ergebnisse der Frequenzerkennung durch mehr korrekte Ergebnisse ausgeglichen werden. Somit erklärt sich die höhere Trefferquote. Die Trefferquoten, Genauigkeiten und durchschnittlichen Abweichungen für die korrekt erkannten Schwebungen sind in Abbildung 6.13 dargestellt.

Die Ergebnisse der Schwebungserkennung über das Gesamtsystem sind sehr ähnlich zu den Testresultaten der reinen Schwebungserkennung. Bei FFT und HSA bestehen nur minimale Unterschiede. Dies gilt sowohl für die Trefferquoten als auch die Genauigkeiten und durchschnittlichen Abweichungen. Lediglich bei der SDF sind stärkere Unterschiede sichtbar. Dort hat das Gesamtsystem mit 84,15 % eine um 2,27 % schlechtere Trefferquote. Bei der Genauigkeit beträgt der Unterschied 2,33 %. Die durchschnittliche Abweichung ist mit 1,33 % ebenfalls 0,06 % höher als bei der reinen Schwebungserkennung.

Beim Gesamtsystem wird die durchgeführte Frequenzerkennung als Grundlage für die Schwebungserkennung verwendet. Aufgrund der etwa 90-prozentigen Trefferquote dieser wäre davon auszugehen gewesen, dass die Trefferquote der Schwebungserkennung ungefähr um den Faktor 0,1 schlechter gegenüber den vorherigen Tests ist. Dass dies nicht der Fall ist, deutet darauf hin, dass bei den Tests der reinen Schwebungserkennung zuvor vor allem die Testfälle fehlgeschlagen sind, bei denen auch die Frequenzerkennung keine korrekten Ergebnisse liefert.



(a) Vergleich der Trefferquoten (links) und Genauigkeiten (rechts).

(b) Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen.

Abbildung 6.13: Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit additiven Testdaten. Zum Vergleich sind die Ergebnisse der Schwebungserkennung aus den Abbildungen 6.9 und 6.10 als dünnere Balken ebenfalls dargestellt.

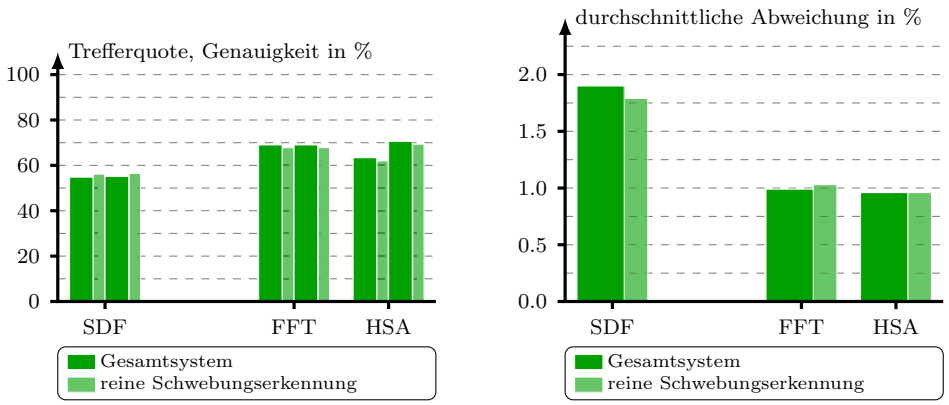
6.3.3 Ergebnisse für GrandOrgue-basierte Testdaten

Die Trefferquote der Frequenzerkennung im Gesamtsystem für die GrandOrgue-basierten Testdaten beträgt 92,13 %. Damit liegt diese, wie bei den additiven Testdaten, über dem Ergebnis der reinen Frequenzerkennung (89,56 %). Dabei ist hier der Abstand mit 2,57 % sogar höher als bei den additiven Testdaten (1,28 %). Die Trefferquoten, Genauigkeiten und durchschnittlichen Abweichungen für die korrekt erkannten Schwebungen sind in Abbildung 6.14 dargestellt.

Auch hier sind, wie bei Verwendung der additiven Testdaten, die Ergebnisse des Gesamtsystems ähnlich zu den Ergebnissen der reinen Schwebungserkennung. Die Unterschiede bei den Trefferquoten und Genauigkeiten liegen nur zwischen 1,19 % und 1,4 %. An der Reihenfolge der Ergebnisse ändert sich dadurch nichts. Die höchste Trefferquote liefert weiterhin FFT, gefolgt von HSA und SDF.

Auch die Unterschiede bei den durchschnittlichen Abweichungen sind ähnlich wie bei den additiven Testdaten. Für die HSA-basierte Analyse ist diese unverändert. Bei FFT ist sie mit 0,99 % um 0,04 % geringer, bei SDF mit 1,90 % um 0,11 % höher.

Zusammenfassend kann festgestellt werden, dass sowohl bei den additiven als auch den GrandOrgue-basierten Testdaten keine gravierenden Unterschiede zu finden waren.



(a) Vergleich der Trefferquoten (links) und Genauigkeiten (rechts).

(b) Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen.

Abbildung 6.14: Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit GrandOrgue-basierten Testdaten. Zum Vergleich sind die Ergebnisse der Schwebungserkennung aus den Abbildungen 6.11 und 6.12 als dünnere Balken ebenfalls dargestellt.

Dementsprechend ist das entwickelte Gesamtsystem, als Verbindung der beiden Komponenten zur Frequenz- und Schwebungserkennung, prinzipiell für die Erkennung von Schwebungen geeignet.

6.4 Vergleich der verschiedenen Pitch-Shifting-Algorithmen

6.4.1 Methodik

In diesem Kapitel soll überprüft werden, ob der bei der Erstellung der Testdaten verwendete Pitch-Shifting-Algorithmus (siehe Kapitel 5.2.1) einen Einfluss auf die Ergebnisse hat. Bisher wurden sämtliche Evaluationsschritte mithilfe der per `librosa.effects.pitch_shift` generierten Testdaten (konkret den Testsets 1, 2, 5 und 6) durchgeführt. Für die folgende Überprüfung werden hingegen Testsets verwendet, bei denen für die Generierung die `pyrubberband.pyrb.pitch_shift`-Funktion genutzt wurde.

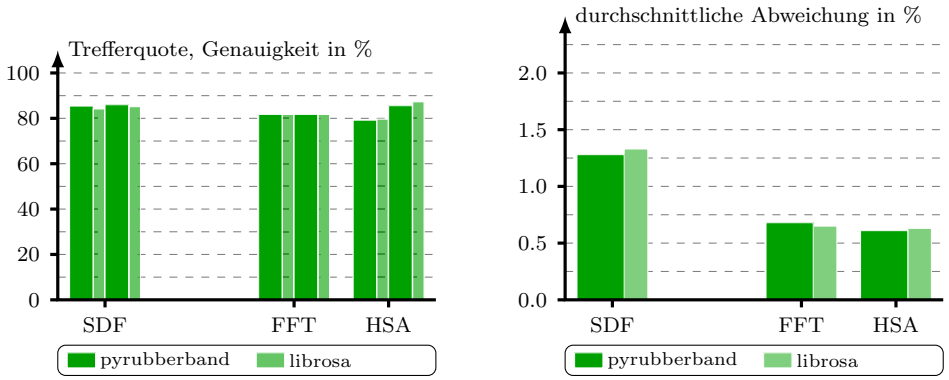
Dazu wird die Evaluation des Gesamtsystems aus dem vorherigen Kapitel 6.3 sowohl für die additiven als auch die GrandOrgue-basierten Testdaten wiederholt und mit den

bisher gefundenen Ergebnissen verglichen. Konkret werden dazu die Testsets *Additive 2 complete* (Nr. 3) und *GrandOrgue 2 complete* (Nr. 7) verwendet.

Sollten dabei keine wesentlichen Unterschiede festgestellt werden, so kann von ausgegangen werden, dass der verwendete Pitch-Shifting-Algorithmus keinen Einfluss auf die Ergebnisse hat. Ist dies nicht der Fall, so muss angenommen werden, dass die ermittelten und verwendeten Parametrierungen für die Frequenz- und Schwebungserkennung zu spezifisch auf die Testsets angepasst wurden und somit für einen Einsatz mit anderen Audiodaten nicht geeignet sind.

6.4.2 Ergebnisse für additive Testdaten

Die Trefferquote der Frequenzerkennung des Gesamtsystems mit den per pyrubberband erstellten, additiven Testdaten beträgt 91,74 %. Dieses Ergebnis liegt damit nah am Ergebnis der per librosa erstellten Testdaten (91,95 %). Die Trefferquoten, Genauigkeiten und durchschnittlichen Abweichungen der Schwebungserkennung sind in Abbildung 6.15 dargestellt.



(a) Vergleich der Trefferquoten (links) und Genauigkeiten (rechts).

(b) Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen.

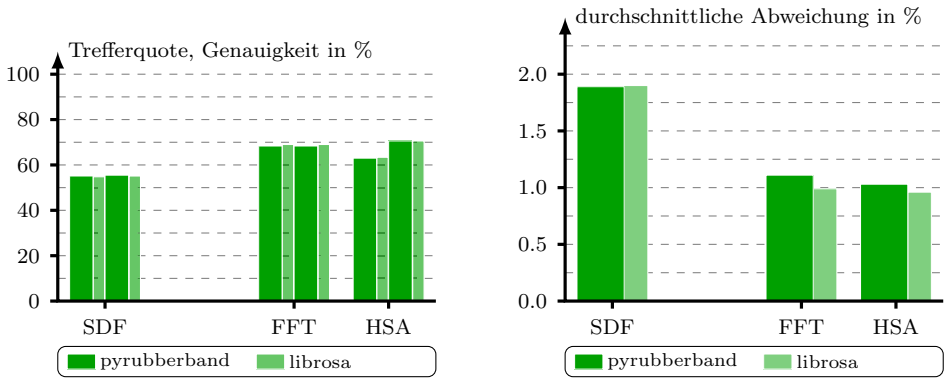
Abbildung 6.15: Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit per pyrubberband erstellten additiven Testdaten. Zum Vergleich sind die Ergebnisse des Gesamtsystems aus Abbildung 6.13 als dünnere Balken ebenfalls dargestellt.

Wie sich dort zeigt, sind kaum Unterschiede zwischen den Ergebnissen der beiden Testsets erkennbar. Bei den Trefferquoten liegen die Unterschiede bei 0,03 % (FFT), 0,46 % (HSA) und 1,27 % (SDF). Das Verhältnis der verschiedenen Verfahren zueinander

bleibt dabei gleich. Die höchste Trefferquote hat weiterhin die SDF, gefolgt von FFT und HSA. Die Unterschiede bei den Genauigkeiten sind mit 0,03 % (FFT), 0,88 % (SDF) und 1,70 % (HSA) ebenfalls relativ gering. Bei den durchschnittlichen Abweichungen sind auch nur kleine Unterschiede zu finden. Diese betragen 0,02 % (HSA), 0,03 % (FFT) und 0,05 % (SDF). Somit kann festgestellt werden, dass die Verwendung eines anderen Pitch-Shifting-Algorithmus für die Erstellung der additiven Testdaten keinen signifikanten Einfluss auf die Ergebnisse hat.

6.4.3 Ergebnisse für GrandOrgue-basierte Testdaten

Die Trefferquote der Frequenzerkennung des Gesamtsystems mit den per pyrubberband erstellten, GrandOrgue-basierten Testdaten beträgt 92,48 %. Dieses Ergebnis liegt damit über den Ergebnissen der per librosa erstellten Testdaten (92,13 %). Die Trefferquoten und durchschnittlichen Abweichungen der Schwebungserkennung sind in Abbildung 6.16 dargestellt.



(a) Vergleich der Trefferquoten (links) und Genauigkeiten (rechts).

(b) Vergleich der durchschnittlichen Abweichungen für korrekt erkannte Schwebungen.

Abbildung 6.16: Vergleich der verschiedenen Konfigurationen des Gesamtsystems mit per pyrubberband erstellten GrandOrgue-basierten Testdaten. Zum Vergleich sind die Ergebnisse des Gesamtsystems aus Abbildung 6.14 als dünnere Balken ebenfalls dargestellt.

Auch hier zeigt der Vergleich ähnliche Ergebnisse für die beiden unterschiedlich generierten Testsets. Bei den Trefferquoten liegen die Unterschiede im Bereich von 0,36 % bis 0,65 %, bei den Genauigkeiten zwischen 0,03 % und 0,65 %. Die durchschnittlichen Abweichungen unterscheiden sich hingegen etwas stärker. Für die SDF ist diese mit 1,89 %

minimal geringer gegenüber 1,90 %. Bei FFT und HSA hingegen sind die Abweichungen mit 1,11 % bzw. 1,03 % bis zu 0,12 % höher als für die per librosa generierten Testdaten.

Zusammenfassend lässt sich feststellen, dass sich die Ergebnisse für die per pyrubberband generierten Testdaten nicht gravierend von den Ergebnissen der per librosa generierten Testdaten unterscheiden. Das gilt sowohl für die additiven als auch die GrandOrgue-basierten Testdaten. Somit sollte die gefundene Konfiguration auch für andere Testdaten geeignet sein.

7 Fazit und Ausblick

7.1 Fazit

Im Rahmen dieser Masterarbeit wurde einen Algorithmus zur Erkennung von akustischen Schwebungen bei gleichzeitig erklingenden Labial- und Lingualpfeifen von Orgeln entwickelt. Diese treten insbesondere beim regelmäßigen Nachstimmen von Lingualpfeifen auf, und werden dabei verwendet, um selbst geringe Frequenzabweichungen wahrnehmen zu können.

Der entwickelte Algorithmus verwendet ein zweistufiges Vorgehen. Im ersten Schritt wird per Frequenzerkennung die Grundfrequenz der Audiodaten ermittelt. Mithilfe dieser wird das Signals dann nach möglichen Schwebungen untersucht. Dazu werden Eigenschaften von diesem ebenfalls über Frequenzerkennungsalgorithmen analysiert. Die verwendeten Eigenschaften können dabei der Verlauf der maximalen Amplituden, die Hüllkurve oder die Verläufe der Fourierkoeffizienten sein.

Für die Evaluation des entwickelten Algorithmus wurden verschiedene Klassen von Testdaten verwendet. Diese wurden sowohl per additiver Synthese als auch mithilfe des Orgelsimulators GrandOrgue generiert. Für die Erzeugung der Schwebungen in den Testsets kamen zwei verschiedene Pitch-Shifting-Algorithmen zum Einsatz. Bei der Evaluation des Algorithmus wurden die Ergebnisse für die verschiedene Frequenzerkennungsverfahren untersucht und miteinander verglichen.

Für die Frequenzerkennung konnte durch den Einsatz des HSA-Verfahrens für alle Testdaten eine Trefferquote von ungefähr 90 % erreicht werden. Schwebungen konnten bei den additiven Daten mit einer Genauigkeit von ca. 85 % erkannt werden. Als Verfahren kam dabei die Analyse der Fourierkoeffizienten per SDF zum Einsatz. Für die GrandOrgue-basierten Testdaten hingegen beträgt die Trefferquote nur knapp 70 %. Dort erfolgte ebenfalls einer Analyse der Fourierkoeffizienten, jedoch per FFT statt SDF.

Basierend auf diesen Ergebnissen kann der in Kapitel 1.2 aufgestellte Hypothese H1 (eine Erkennung ist mit Trefferquoten und Genauigkeiten von mehr als 60 % möglich) zugestimmt werden. Mit je 85 % und 70 % wurde dieser Schwellwert sowohl für die additiven als auch die GrandOrgue-basierten Testdaten übertroffen.

Die weitere Hypothese H2 besagte, dass die Verwendung von Frequenzerkennungsverfahren aus dem Frequenzbereich die besseren Ergebnisse im Vergleich zu den Verfahren

aus dem Zeitbereich liefert. Bei der Frequenzerkennung trifft dies, sowohl für die additiven als auch die GrandOrgue-basierten Testdaten, zu. Dort liefert die Verwendung der HSA als Verfahren aus dem Frequenzbereich die besten Ergebnisse. Bei der Schwebungserkennung aber ist dies jedoch nicht immer der Fall. Für die GrandOrgue-basierten Testdaten liefern zwar mit der Verwendung der FFT und HSA zwei Verfahren aus dem Frequenzbereich die besten Ergebnisse. Bei den additiven Testdaten hingegen werden die besseren Ergebnisse bei der SDF erreicht. Dementsprechend kann der Hypothese H2 nur bedingt zugestimmt werden.

Entsprechend der letzten Hypothese H3 wäre es möglich, ein Stimmgerät auf Basis des entwickelten Algorithmus zu entwerfen. Dazu soll sowohl die Audio- und Schwebungsfrequenz als auch eine Stimmrichtung angezeigt werden. Die ersten beiden Informationen können direkt mithilfe des Algorithmus berechnet werden. Für die Bestimmung der Stimmrichtung wäre allerdings zusätzliche Implementierungsarbeit notwendig. Grund dafür ist, dass bei dem Auftreten einer Schwebung nicht bekannt ist, ob die Lingualpfeife der höheren oder niedrigeren der beiden berechneten Ausgangsfrequenzen entspricht.

Um diese Information zu erhalten, könnten während des Stimmens einer kompletten Orgel die Frequenzen der zuvor gestimmten Töne (die implizit den Frequenzen der Labialpfeifen entsprechen) zwischengespeichert werden. Mithilfe dieser Werte wäre es möglich, die erwartete Frequenz zu berechnen und mit den beiden zur Frage stehenden Frequenzen zu vergleichen. Die näher daran liegende Frequenz würde dann der Labialpfeife und somit der Zielfrequenz entsprechen.

Alternativ würde auch die Möglichkeit bestehen, die Veränderung der Frequenzen während eines konkreten Stimmvorgangs zu beobachten. Dabei sollte sich die gemessene resultierende Frequenz mit kleiner werdender Schwebung in Richtung der korrekten Frequenz verschieben und am Ende mit dieser übereinstimmen. Eine der beiden berechneten Ausgangsfrequenzen verändert sich während des Stimmens doppelt so schnell. Diese entspricht der gestimmten Lingualpfeife. Die Frequenz der Labialpfeife, und damit die Zielfrequenz, hingegen bleibt während des Stimmvorgangs unverändert. Über die Beobachtung der Veränderungen wäre es somit auch möglich, die Zielfrequenz und somit eine Stimmrichtung zu ermitteln.

Somit kann der Hypothese H3 ebenfalls zugestimmt werden, mit der Einschränkung, dass dafür zusätzliche Entwicklungsarbeit notwendig wäre.

7.2 Ausblick

Eines der größten Probleme des Algorithmus ist, dass bei der Verwendung von realitätsnäheren Testdaten (den GrandOrgue-basierten Testdaten) schlechtere Ergebnisse

erreicht werden. Die Evaluation mit den additiven Testdaten hat allerdings gezeigt, dass Trefferquoten von über 85 % für die Schwebungserkennung möglich sind. In weiteren Untersuchungen könnte analysiert werden, wie diese Trefferquoten auch für die GrandOrgue-basierten Testdaten erreicht werden könnten. Dazu könnten verschiedene Möglichkeiten zur Vorverarbeitung des Audiosignals sowie der Einfluss von Störgeräuschen und Rauschen untersucht werden. Ebenso könnten die Tests mit realen Audioaufnahmen wiederholt werden, um zu Trefferquoten für diese Testdaten zu ermitteln.

Literaturverzeichnis

[AEVS19] Arnie Gedanken Branta, Emanuel Marques Martins, Verônica Isabela Quandt and Solivan Arantes Valente: *Development of a triple input musical instrument tuner using Yin algorithm*, *International Journal of Advanced Engineering Research and Science*, vol. 6(12):pp. 380–384, Dec. 2019, ISSN N: 2349-649.

[AH02] Alain de Cheveigné and Hideki Kawahara: *YIN, a fundamental frequency estimator for speech and music*, *The Journal of the Acoustical Society of America*, vol. 111(4):pp. 1917–1930, 2002.

[Bri15] Brian McFee: *pyrubberband.pyrb.pitch_ shift*, 2015.
URL `https://pyrubberband.readthedocs.io/en/stable/generated/pyrubberband.pyrb.pitch_shift.html` [Accessed: 2022-09-27]

[CAX09] Chunghsin Yeh, Axel Roebel and Xavier Rodet: *Multiple Fundamental Frequency Estimation and Polyphony Inference of Polyphonic Music Signals*, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18(6):pp. 1116–1126, Aug. 2009, ISSN 1558-7916.

[DD08] David L. Olson and Dursun Delen: *Advanced Data Mining Techniques*, Springer-Verlag, Berlin Heidelberg, 2008, ISBN 978-3-540-76917-0.

[Dira] Dirk's Projects: *Dirk's Stimmgerät für chromatische Instrumente v1.3 - Handbuch*.
URL `https://www.dirksprojects.nl/index.php?Lan=german&Page=Tuner/Manual13.php` [Accessed: 2021-10-24]

[Dirb] Dirk's Projects: *Professionelles Stimmgerät für Akkordeons v2.3 - Handbuch*.
URL `https://www.dirksprojects.nl/index.php?Lan=german&Page=Tuner/Manual22A.php` [Accessed: 2022-07-29]

[FDD+03] Frank-Michael Becker, Dr. Hubert Bossek, Dr. Lutz Engelmann, Dr. Christine Ernst, Dr. habil. Günter Fanghänel, Heinz Höhne, Dr. Astrid Kalenberg, Rudi Lenertat, Manuela Liesenberg, Rainer Löffler, Dr. Günter Liesenberg,

Prof. Dr. habil. Lothar Meyer, Doz. Dr. habil. Christa Pews-Hocke, Dr. habil. Bernd Raum, Dr. Gerd-Dietrich Schmidt, Dr. Peter Seidel, Helga Simon, Dr. habil. Reinhard Stamm, Prof. Dr. habil. Karlheinz Weber and Dr. Adria Wehser: *Formelsammlung bis zum Abitur*, paetec Gesellschaft für Bildung und Technik mbH, Berlin, 1. auflage edn., 2003, ISBN 978-3-89818-700-8.

[Fra16] Frank Nielsen: *Hierarchical clustering*, in *Introduction to HPC with MPI for Data Science*, pp. 195–211, Feb. 2016, ISBN 978-3-319-21902-8.

[Her77] Hermann von Helmholtz: *Die Lehre von den Tonempfindungen als physiologische Grundlage für die Theorie der Musik*, Freidrich Vieweg und Sohn, Braunschweig, 1877.

[Joa15] Joachim Stange-Elbe: *Computer und Musik: Grundlagen, Technologien und Produktionsumgebungen der digitalen Musik*, De Gruyter Oldenbourg, 2015, ISBN 978-3-486-72019-8.

[JPM14] Jochen Kuhn, Patrik Vogt and Michael Hirth: *Analyzing the acoustic beat with mobile devices*, *The Physics Teacher*, vol. 52(4):pp. 248–249, Apr. 2014.

[Jü07] Jürgen Eichler: *Physik: Grundlagen für das Ingenieurstudium – kurz und prägnant*, Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH, Wiesbaden, 3., überarbeitete und ergänzte auflage edn., 2007, ISBN 978-3-8348-0223-1.

[LG] Lars Palo and Graham Goode: *Bureå church organ sampleset*.
URL `https://familjenpalo.se/vpo/about-burea-church/` [Accessed: 2021-11-15]

[lib13] librosa development team: *librosa.effects.pitch_ shift*, 2013.
URL `https://librosa.org/doc/latest/generated/librosa.effects.pitch_shift.html` [Accessed: 2022-09-27]

[LMAM76] Lawrence R. Rabine, Michael J. Cheng, Aaron E. Rosenberg and Carol A. McGonega: *A Comparative Performance Study of Several Pitch Detection Algorithms*, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24(5):pp. 399 – 418, Oct. 1976, ISSN 0096-3518.

[MA09] Mary Lourde Regeena and Anjali Kuppayil Saji: *A Digital Guitar Tuner*, *International Journal of Computer Science and Information Security*, vol. 6(2):pp. 82–88, Dec. 2009, ISSN 1947-5500.

[Mar] Markus Gorski: *Die Frequenzverhältnisse*.
URL `https://www.lehrklaenge.de/PHP/Tonsystem/Frequenzverhaeltnisse.php` [Accessed: 2022-07-20]

[MAS03] Mürsel Önder, Aydın Akan and Semih Bingöl: *Pitch detection for monophonic musical notes*, in *Third International Conference on Electrical and Electronic Engineering (ELECO)*, vol. 1, Citeseer, 2003.

[Mat] MathWorks: *Envelope Detection*.
URL `https://de.mathworks.com/help/dsp/ug/envelope-detection.html` [Accessed: 2022-03-31]

[PAC01] Patricio de la Cuadra, Aaron Master and Craig Sapp: *Efficient Pitch Detection Techniques for Interactive Music*, in *In Proceedings of the 2001 International Computer Music Conference, La Habana*, vol. 2001, La Habana, 2001.

[PG05] Philip McLeod and Geoff Wyvill: *A smarter way to find pitch*, Barcelona, Spain, 2005.

[Pyt22] Python Software Foundation: *multiprocessing — Process-based parallelism*, 2022.
URL `https://docs.python.org/3/library/multiprocessing.html#module-multiprocessing` [Accessed: 2022-10-25]

[RJ08] Rafael George Amado and Jozue Vieira Filho: *Pitch Detection Algorithms Based on Zero-Cross Rate and Autocorrelation Function for Musical Notes*, in *2008 International Conference on Audio, Language and Image Processing*, pp. 449–454, IEEE, Shanghai, China, 2008, ISBN 978-1-4244-1723-0.

[SS11] Steffen Goebbels and Stefan Ritter: *Mathematik verstehen und anwenden*, Spektrum Akademischer Verlag, Heidelberg, 2011, ISBN 978-3-8274-2761-8.

[Ste78] Stephanie Seneff: *Real-time harmonic pitch detector*, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26(4):pp. 358 – 365, Aug. 1978, ISSN 0096-3518.

[SU15] Sebastian Kraft and Udo Zölzer: *Polyphonic pitch detection by matching spectral and autocorrelation peaks*, in *2015 23rd European Signal Processing*

Conference (EUSIPCO), pp. 1301–1305, IEEE, Nice, France, 2015, ISBN 978-0-9928626-3-3.

[TI18] Thomas A. Goodman and Ian Batten: *Real-Time Polyphonic Pitch Detection on Acoustic Musical Signals*, in *2018 IEEE International Symposium on Signal Processing and Information Technology*, pp. 656–661, IEEE, Louisville, KY, USA, 2018, ISBN 978-1-5386-7568-7.

[USLR20] Urja Kulkarni, Shrishti Kaushik, Lanita Lobo and Reena Sonkusare: *Comparative Study of Digital Signal Processing Techniques for Tuning an Acoustic Guitar*, in *2020 7th International Conference on Smart Structures and Systems (ICSSS)*, IEEE, Chennai, India, 2020, ISBN 978-1-72817-223-1.

[Wei] Weiblen Spezialwerkzeuge: *Messingstimmeisen*.
URL `https://www.weiblen.de/fileadmin/_migrated/pics/04600_Stimmeisensatz.jpg` [Accessed: 2022-04-02]

[Wol91] Wolfgang Adelung: *Einführung in den Orgelbau*, Breitkopf & Härtel, Wiesbaden, 2., überarbeitete und erweiterte ausgabe edn., 1991, ISBN 3-7651-0279-2.

[YD07] Yipeng Li and DeLiang Wang: *Pitch Detection in Polyphonic Music using Instrument Tone Models*, in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, pp. 481–484, IEEE, Honolulu, HI, USA, 2007, ISBN 1-4244-0727-3.

A Anhang

A.1 Implementierung der verwendeten Algorithmen

Dieses Kapitel gibt einen Überblick über die Implementierung der verwendeten Algorithmen. Dies umfasst den Clustering-Algorithmus (siehe Kapitel A.1.1), den Algorithmus zur Verbesserung der FFT-Ergebnisse (siehe Kapitel A.1.2) sowie die einzelnen Frequenzerkennungsverfahren (siehe Kapitel A.1.3) ein.

A.1.1 Clustering-Algorithmus

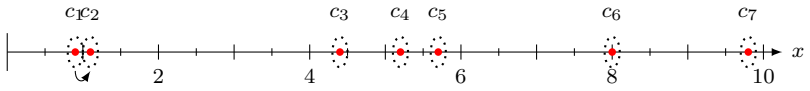
Der Clustering-Algorithmus wird dafür verwendet, nah beieinanderliegende Ergebnisse bzw. Werte in gemeinsame Gruppen zusammenzufassen. Er findet Anwendung bei der Berechnung der HSA (vgl. Kapitel 3.2.2) und bei der Zusammenfassung der Ergebnisse von mehreren Frequenzerkennungen (vgl. Kapitel 4.2) sowie der einzelnen Schwebungsfrequenzen bei der Analyse der Fourierkoeffizienten (vgl. Kapitel 4.3.3).

Für die Berechnung wird ein hierarchischer agglomerativer Ansatz gewählt (siehe Abbildung A.1). Das bedeutet, zu Beginn existieren viele kleine Cluster C_i mit nur einem Element, die später iterativ zu größeren Clustern zusammengefasst werden. Das finale Ergebnis liegt vor, wenn keine Cluster mehr zusammengefasst werden können.

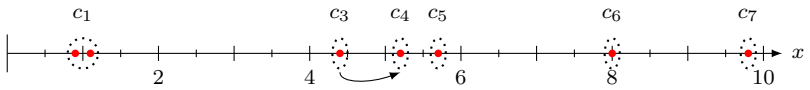
Jedem existierenden Cluster C_i wird durch die Wertefunktion $W(C_i)$ ein numerischer Wert zugewiesen. Dafür wird das arithmetische Mittel aller enthaltenen Werte $C_{i,k}$ verwendet:

$$W(C_i) = \frac{\sum_{j=0}^{\|C_i\|} C_{i,j}}{\|C_i\|}. \tag{A.1}$$

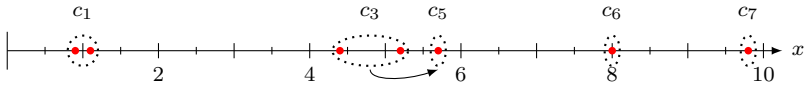
Für die Beurteilung, ob und welche Klassen zusammengefasst werden können, muss außerdem das Distanzmaß $D(C_i, C_j)$ zwischen zwei Clustern C_i und C_j definiert werden. Da alle hier betrachteten Werte eindimensional sind, könnte dafür der Abstand bzw. die Differenz zwischen beiden verwendet werden. Aufgrund des exponentiellen Zusammenhangs zwischen Tonhöhen und Frequenzen (vgl. Kapitel 2.1.3) bietet sich hingegen eine



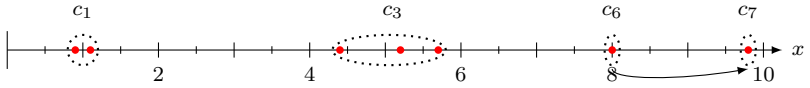
(a) Initialer Zustand. Hier können die Cluster c_1 und c_2 zusammengefasst werden ($D(c_1, c_2) \approx 0{,}22$).



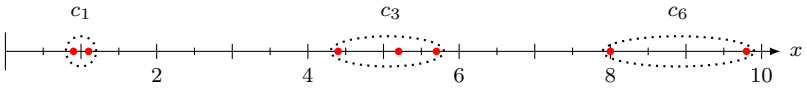
(b) Zustand nach dem ersten Schritt. Hier können die Cluster c_3 und c_4 zusammengefasst werden ($D(c_3, c_4) \approx 0{,}18$).



(c) Zustand nach dem zweiten Schritt. Hier können die Cluster c_3 und c_5 zusammengefasst werden ($D(c_3, c_5) \approx 0{,}19$).



(d) Zustand nach dem dritten Schritt. Hier können die Cluster c_6 und c_7 zusammengefasst werden ($D(c_6, c_7) \approx 0{,}23$).



(e) Finaler Zustand nach dem vierten Schritt.

Abbildung A.1: Beispielhafter Ablauf des Clustering-Algorithmus mit sieben Clustern. Als Wert für $D_{\max}$ wurde 0,25 verwendet.

Betrachtung der relativen Abstände, jeweils in Bezug auf den kleineren der beiden Werte, an:

$$D(C_i, C_j) = \begin{cases} \dfrac{W(C_i)}{W(C_j)} - 1 & \text{falls } C_i \geq C_j \\ \dfrac{W(C_j)}{W(C_i)} - 1 & \text{falls } C_i < C_j \end{cases} . \tag{A.2}$$

$D(C_i, C_j)$ hat immer einen positiven Wert. Die Entscheidung, ob Cluster zusammengefasst werden sollen oder nicht, kann anhand des Distanzmaßes getroffen werden. Ist dessen Wert kleiner als oder gleich groß wie ein festgelegtes Maximum $D_{\max}$, so werden die beiden Cluster verbunden. Ist dies nicht der Fall, so verbleiben sie getrennt.

Dieses Vorgehen steht im Gegensatz zu den üblicherweise verwendeten Algorithmen, wie sie z.B. in [Fra16] beschrieben sind. Dort werden in jedem Schritt die zwei Cluster

mit der geringsten Distanz zusammengefasst, bis nur noch ein einziges Cluster, das alle Elemente enthält, übrig ist. Als Ergebnis erhält man eine grafische Repräsentation der nacheinander stattgefundenen Zusammenfassungen als Binärbaum, das sogenannte Dendrogramm. Die Verwendung dieses Vorgehens ist jedoch nicht sinnvoll, da nicht einfach bestimmt werden kann, welcher der stattgefundenen Zwischenschritte der optimalen Klassifizierung entspricht. Auch der Einsatz des k-Means-Algorithmus ist nicht möglich, da zuvor nicht bekannt ist, wie viele Cluster bzw. ähnliche Werte in den Daten vorliegen.

Der Pseudocode des verwendeten Clustering-Algorithmus ist in Algorithmus A.1 gegeben. In diesem werden die Cluster als Liste von Listen repräsentiert. Die äußere Liste enthält dabei alle gefundenen Cluster. Die inneren Listen entsprechenden den einzelnen Clustern und enthalten jeweils alle Werte, die zu diesen gehören. Für den Pseudocode wird vorausgesetzt, dass die verwendete Listen-Implementierung die folgenden Funktionen unterstützt:

- createList(): Erstellt eine neue leere Liste und gibt diese zurück.
- size(L): Gibt die Größe der gegebenen Liste L zurück.
- append(L, e): Fügt das Element bzw. den Wert e ans Ende der Liste L an.
- remove(L, i): Entfernt das Element mit dem Index i aus der Liste L.
- sort(L): Sortiert alle Elemente der Liste L aufsteigend.
- $L[i]$: Gibt das i-te Element der Liste L zurück.

Algorithmus A.1 Clustering-Algorithmus.

Require: $V[0 \,..\, N-1] \in \mathbb{R}, \quad V[i] \leq V[i+1] \; \forall i \in [0, N-2], \quad D_{\max} \in \mathbb{R}, \quad o \in \{\text{true}, \text{false}\}$

```
1: procedure executeClustering(V[0 .. N − 1], D_max, o)
2:     C ← initClusters(V)
3:     tryMergeAllClusters(C, D_max, false)
4:     tryMergeAllClusters(C, D_max, o)
5:     return C
6: end procedure
```

Der Algorithmus erhält die folgenden Parameter:

- V: Array mit den N Werten, auf die das Clustering angewendet werden soll.

- $D_{\max}$: maximale Distanz, die zwischen zwei Clustern vorliegen darf, um diese zusammenfassen zu können.

Als Ergebnis wird die resultierende Liste von Clustern C zurückgegeben.

Im ersten Schritt werden die initialen Cluster C erstellt (siehe Algorithmus A.2). Dazu wird für jeden Wert ein gesondertes Cluster erstellt, das jeweils nur diesen einen Wert enthält. Die aufsteigende Sortierung der Originalwerte V bleibt auch bei den erstellten Clustern C erhalten.

Algorithmus A.2 Algorithmus zum Initialisieren der Cluster.

Require: $V[0 \,..\, N-1] \in \mathbb{R}$
1: **procedure** INITCLUSTERS($V[0 \,..\, N-1]$)
2: $C \leftarrow$ CREATELIST()
3: **for each** $v \in V$ **do**
4: $l \leftarrow$ CREATELIST()
5: APPEND(l, v)
6: APPEND(C, l)
7: **end for**
8: **return** C
9: **end procedure**

Im nächsten Schritt werden die Cluster so lange iterativ zusammengefasst, bis keine Veränderung mehr stattfindet. Dabei werden alle Cluster mit ähnlichen Werten zusammengefasst. Danach können, falls über den Parameter o indiziert, diejenigen Werte, die Vielfache voneinander sind, zusammengefasst werden. Dort wird dann nicht die Distanz D der Cluster betrachtet, sondern ausschließlich die Nachkommastelle der Distanz. Ist deren Wert kleiner als $D_{\max}$, so werden die Cluster zusammengefasst. Das Zusammenfassen erfolgt in beiden Schritten über die Funktion TRYMERGEALLCLUSTERS() (siehe Algorithmus A.3.

Diese verwendet das Flag f, um festzustellen, ob im vorherigen Durchlauf Änderungen stattfanden oder nicht. Dieses wird jeweils mit 0 initialisiert und bei jeder Änderung auf 1 gesetzt. In jedem Schleifendurchlauf werden nacheinander die Distanzen $D(C_i, C_j)$ zwischen allen Clustern C_i und C_j berechnet. Dabei wird zwischen den beiden Fällen (Parameter o = true bzw. o = false) unterschieden. Wird eine Distanz gefunden, die kleiner als der festgelegte Maximalwert $D_{\max}$ ist, so werden die beiden Cluster C_i und C_j zusammengefasst (siehe Algorithmus A.4). Dazu werden alle Elemente des Clusters C_j dem Cluster C_i hinzugefügt. Sind die Cluster keine Vielfachen voneinander (der Parameter o ist false), so werden die Werte direkt übernommen. Im entgegensetzen Fall

Algorithmus A.3 Algorithmus zum Zusammenfassen aller Cluster.

Require: $C \in \text{List}(\text{List}(\mathbb{R}))$, $D_{\max} \in \mathbb{R}$, $o \in \{\text{true}, \text{false}\}$

```
1: procedure tryMergeAllClusters(C, D_max, o)
2:     f ← 1
3:     while f ≠ 0 do
4:         f ← 0
5:         for each i ∈ [0, size(C) − 2] do
6:             for each j ∈ [i + 1, size(C) − 1] do
7:                 if (o = false ∧ D(C[i], C[j]) ≤ D_max) ∨ (o = true ∧ D(C[i], C[j])
   mod 1 ≤ D_max) then
8:                     mergeClusters(C, i, j)
9:                     f ← 1
10:                    break
11:                end if
12:            end for
13:        end for
14:    end while
15:    return C
16: end procedure
```

werden die Werte vor dem Hinzufügen um den Faktor o_f, der dem Verhältnis entspricht, skaliert, um später falsche Ergebnisse bei der Berechnung der Distanzen zu vermeiden. Danach wird das Cluster C_j aus der Menge der gefundenen Cluster entfernt und somit in den folgenden Iterationen nicht weiter betrachtet.

Die tryMergeAllClusters()-Funktion terminiert jeweils, wenn in einem Schleifendurchlauf keine Veränderungen mehr an den Clustern stattfinden. Dies ist der Fall, wenn keine der Distanzen mehr kleiner als $D_{\max}$ ist. Dann wird auch das Flag f nicht mehr auf 1 gesetzt und folglich die äußerste while-Schleife verlassen. Abschließend wird die finale Liste der gefundenen Cluster C zurückgegeben.

A.1.2 Interpolationsalgorithmus für FFT-basierte Ergebnisse

Bei der Verwendung von FFT-basierten Frequenzerkennungsverfahren wird die folgende Formel für die Berechnung der Frequenz f eines Bins i verwendet:

$$f = i \cdot \frac{f_S}{W} \quad , i \in \left[0, \frac{W}{2}\right]. \tag{A.3}$$

Algorithmus A.4 Algorithmus zum Zusammenfassen von zwei Clustern.

Require: $C \in \text{List}(\text{List}(\mathbb{R}))$, $i_1 \in [0, \text{SIZE}(C) - 1]$, $i_2 \in [0, \text{SIZE}(C) - 1]$, $i_1 \neq i_2$, $o \in \{\text{true}, \text{false}\}$

```
1: procedure MERGECLUSTERS(C, i1, i2, o)
2:     of ≈ W(C[i2]) ÷ W(C[i1])
3:     for each v ∈ C[i2] do
4:         if o = true then
5:             APPEND(C[i1], v ÷ of)
6:         else
7:             APPEND(C[i1], v)
8:         end if
9:     end for
10:    REMOVE(C, i2)
11: end procedure
```

Dabei bezeichnet f_S die Samplerate des Signals und W die verwendete Fenstergröße. Mit diesem Ansatz kann eine Genauigkeit von $\frac{f_S}{W}$ erreicht werden. Bei der verwendeten Samplerate f_S von 44,1 kHz und der Fenstergröße W von 8.192 führt dies folglich zu einer Auflösung von ca. 5,38 Hz.

Für die Frequenzerkennung, also die Suche des globalen Maximums im Amplitudenspektrum, wäre jedoch eine höhere Genauigkeit wünschenswert. Prinzipiell könnte diese durch Erhöhen der Fenstergröße erreicht werden. Aufgrund des indirekt proportionalen Zusammenhangs zwischen Fenstergröße und Auflösung wäre allerdings eine entsprechend starke Erhöhung notwendig. Für eine Genauigkeit von 1 Hz müsste die Fenstergröße auf 44.100 (entspricht einer Sekunde Audiodaten), für eine Genauigkeit von 0,5 Hz sogar auf 88.200 (entspricht zwei Sekunden Audiodaten), vergrößert werden.

Eine alternative Möglichkeit stellt die Verwendung einer quadratischen Interpolation, wie sie u.a. in [USLR20] und [AH02] beschrieben wird, dar. Zur Verbesserung der Genauigkeit wird nicht die Gesamtauflösung der FFT erhöht, sondern nur der gefundene Index $i_{\max}$, basierend auf dessen umgebenden Werten, interpoliert (siehe Abbildung A.2).

Dafür werden die beiden Indizes $i_{\max} - 1$ und $i_{\max} + 1$ zusätzlich mit betrachtet. Somit ergeben sich die folgenden drei Punkte:

- $P_1 = (x_1, y_1) = (i_{\max} - 1, S_{\text{FFT}}[i_{\max} - 1])$
- $P_2 = (x_2, y_2) = (i_{\max}, S_{\text{FFT}}[i_{\max}])$
- $P_3 = (x_3, y_3) = (i_{\max} + 1, S_{\text{FFT}}[i_{\max} + 1])$.

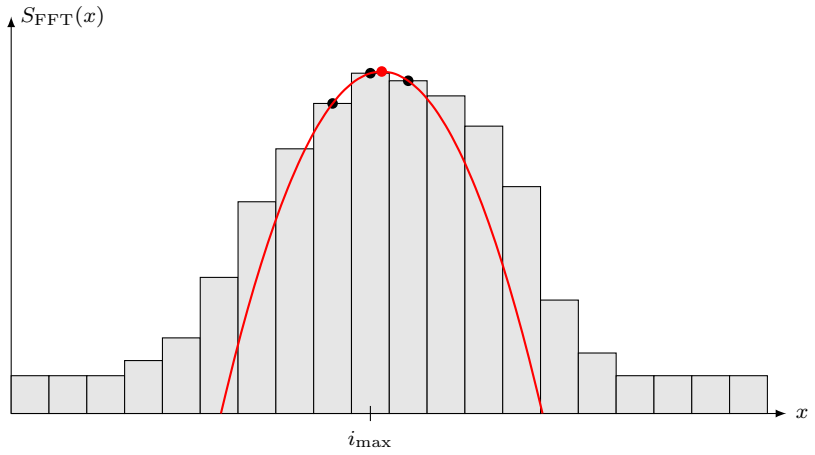


Abbildung A.2: Berechnung des FFT-Maximums per Interpolation. Dargestellt sind die für die Interpolation verwendeten FFT-Werte (schwarz) sowie die daraus interpolierte Funktion und deren Maximum (rot).

Durch diese Punkte kann eine quadratische Funktion der Form $y = a \cdot x^2 + b \cdot x + c$ gelegt werden. Deren Extremstelle x_{ex} kann dann allgemein über die folgende Formel berechnet werden[1]:

$$x_{\mathrm{ex}} = -\frac{b}{2 \cdot a}. \tag{A.6}$$

Für die Bestimmung der Parameter a, b und c aus den Punkten P_1, P_2 und P_3 kann ein lineares Gleichungssystem aufgestellt werden:

$$\left| \begin{array}{rcrcrcl} y_1 & = & a \cdot x_1^2 & + & b \cdot x_1 & + & c \\ y_2 & = & a \cdot x_2^2 & + & b \cdot x_2 & + & c \\ y_3 & = & a \cdot x_3^2 & + & b \cdot x_3 & + & c \end{array} \right|. \tag{A.7}$$

[1] Für die erste Ableitung von $f(x) = a \cdot x^2 + b \cdot x + c$ ergibt sich folgendes:

$$f'(x) = 2 \cdot a \cdot x + b. \tag{A.4}$$

Entsprechend der notwendigen Bedingung für Extremstellen ($f'(x) = 0$) lässt sich die folgende Berechnung aufstellen:

$$0 = 2 \cdot a \cdot x_{\mathrm{ex}} + b \quad \Longleftrightarrow \quad x_{\mathrm{ex}} = -\frac{b}{2 \cdot a}. \tag{A.5}$$

Da es sich bei A.4 um eine lineare Funktion handelt, besitzt diese bei x_{ex} einen Vorzeichenwechsel. Somit ist auch eine hinreichende Bedingung für Extrema erfüllt.

Mithilfe des Gauß-Jordan-Algorithmus ergeben sich die folgenden Formeln zur Berechnung der Parameter a, b und c[2]:

$$a = \frac{(y_3 - y_1) \cdot (x_2 - x_1) - (y_2 - y_1) \cdot (x_3 - x_1)}{(x_3^2 - x_1^2) \cdot (x_2 - x_1) - (x_2^2 - x_1^2) \cdot (x_3 - x_1)} \tag{A.9}$$

$$b = \frac{y_2 - y_1 - a \cdot (x_2^2 - x_1^2)}{x_2 - x_1} \tag{A.10}$$

$$c = y_1 - x_1 \cdot b - x_1^2 \cdot a. \tag{A.11}$$

Mithilfe dieser kann der interpolierte Index i_{max2} berechnet werden. Der entsprechende Pseudocode zur Funktion interpolateFftIndex() ist in Algorithmus A.5 gegeben.

Algorithmus A.5 Algorithmus zur Interpolation von FFT-Ergebnissen.

Require: $S_{\text{FFT}}[0 \,..\, N-1] \in \mathbb{R}, \quad i_{\max} \in [0, N-1]$

procedure interpolateFftIndex($S_{\text{FFT}}[0 \,..\, N-1]$, $i_{\max}$)
    `// Randfälle prüfen`
    if $i_{\max} = 0 \vee i_{\max} = N - 1$ **then return** $i_{\max}$ **end if**

    `// Parameter a und b berechnen`
    $x_1 \leftarrow i_{\max} - 1, \quad x_2 \leftarrow i_{\max}, \quad x_3 \leftarrow i_{\max} + 1$
    $y_1 \leftarrow S_{\text{FFT}}[x_1], \quad y_2 \leftarrow S_{\text{FFT}}[x_2], \quad y_3 \leftarrow S_{\text{FFT}}[x_3]$
    $a \leftarrow \frac{(y_3-y_1)\cdot(x_2-x_1)-(y_2-y_1)\cdot(x_3-x_1)}{(x_3^2-x_1^2)\cdot(x_2-x_1)-(x_2^2-x_1^2)\cdot(x_3-x_1)}$
    $b \leftarrow \frac{y_2-y_1-a\cdot(x_2^2-x_1^2)}{x_2-x_1}$

    `// interpoliertes Maximum berechnen`
    $i_{\text{max2}} \leftarrow (-b) \div (2 \cdot a)$
    return i_{max2}
end procedure

[2]

$$\left(\begin{array}{ccc|c} 1 & x_1 & x_1^2 & y_1 \\ 1 & x_2 & x_2^2 & y_2 \\ 1 & x_3 & x_3^2 & y_3 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & x_1 & x_1^2 & y_1 \\ 0 & x_2 - x_1 & x_2^2 - x_1^2 & y_2 - y_1 \\ 0 & x_3 - x_1 & x_3^2 - x_1^2 & y_3 - y_1 \end{array}\right) \Rightarrow$$
$$\left(\begin{array}{ccc|c} 1 & x_1 & x_1^2 & y_1 \\ 0 & x_2 - x_1 & x_2^2 - x_1^2 & y_2 - y_1 \\ 0 & 0 & \begin{array}{l}(x_3^2 - x_1^2) \cdot (x_2 - x_1) - \\ (x_2^2 - x_1^2) \cdot (x_3 - x_1)\end{array} & \begin{array}{c}(y_3 - y_1) \cdot (x_2 - x_1) - \\ (y_2 - y_1) \cdot (x_3 - x_1)\end{array} \end{array}\right) \tag{A.8}$$

A.1.3 Frequenzerkennungsalgorithmen

Zero-Crossing Rate

Der Pseudocode für die Frequenzerkennung per ZCR ist in Algorithmus A.6 gegeben. An die entsprechende Funktion pitchDetectionByZcr() werden die folgenden Parameter übergeben:

- S: Audiodaten, die analysiert werden sollen.
- f_S: Samplerate der Audiodaten.

Als Ergebnis wird die ermittelte Frequenz f zurückgegeben.

Algorithmus A.6 Algorithmus zur Frequenzerkennung per ZCR.

Require: $S[0 \,..\, N-1] \in \mathbb{R}, \quad f_S \in \mathbb{N}$
1: **procedure** pitchDetectionByZcr($S[0 \,..\, N-1]$, f_S)
2: `// ZCR berechnen`
3: $L, i_{\text{start}} \leftarrow$ initZcr(S)
4: $z_{\text{count}} \leftarrow 0, \quad i_{\text{first}} \leftarrow -1, \quad i_{\text{last}} \leftarrow -1, \quad v_{\text{last}} \leftarrow S[i_{\text{start}}]$
5: **for each** $i \in [i_{\text{start}}, N-1]$ **do**
6: **if** $((v_{\text{last}} < L) \wedge (S[i] > L)) \vee ((v_{\text{last}} > -1 \cdot L) \wedge (S[i] < -1 \cdot L))$ **then**
7: $z_{\text{count}} \leftarrow z_{\text{count}} + 1$
8: **if** $i_{\text{first}} = -1$ **then** $i_{\text{first}} \leftarrow i$ **end if**
9: $i_{\text{last}} \leftarrow i$
10: $v_{\text{last}} \leftarrow S[i]$
11: **end if**
12: **end for**

13: `// Frequenz berechnen`
14: $zcr = (z_{\text{count}} - 1) \div (i_{\text{last}} - i_{\text{first}})$
15: $f = (zcr \cdot f_s) \div 2$
16: **return** f
17: **end procedure**

Zur Steigerung der Genauigkeit wird für die Berechnung ein zuvor berechneter Schwellwert L verwendet. Außerdem werden die exakten Positionen des ersten und letzten gefundenen Nulldurchlaufs gespeichert und in die Berechnung der Frequenz mit einbezogen. Somit wird die Genauigkeit erhöht, da die Größe der zugrundeliegenden Daten exakt bekannt ist.

Im ersten Schritt werden alle benötigten Initialisierungen vorgenommen. Dazu wird in der Funktion initZcr(S) der Schwellwert L als 1,2-faches des Durchschnitts aller absoluten Werte berechnet (siehe Algorithmus A.7). Zusätzlich wird der Index i_{start} gesucht. Dieser entspricht der ersten Position in den Daten, deren Wert nicht im Intervall $[-L, L]$, und damit innerhalb der Schwellwerte, liegt. An dieser Position wird im nächsten Schritt das Zählen der Nulldurchläufe gestartet. Außerdem werden die folgenden Variablen initialisiert:

- z_{count}: Variable für die Anzahl der gefundenen Nulldurchläufe.
- i_{first}: Variable für den Index, an dem der erste Nulldurchlauf gefunden wurde.
- i_{last}: Variable für den Index, an dem der letzte Nulldurchlauf gefunden wurde.
- v_{last}: Variable für den Signalwert beim letzten gefundenen Nulldurchlauf.

Algorithmus A.7 Algorithmus zur Initialisierung der ZCR-Berechnung.

Require: $S[0 \,..\, N-1] \in \mathbb{R}$
1: **procedure** initZcr($S[0 \,..\, N-1]$)
2: $L \leftarrow 1{,}2 \cdot \frac{1}{N} \cdot \sum_{i=0}^{N-1} |S[i]|$
3: $i_{\text{start}} \leftarrow 0$
4: **while** $S[i_{\text{start}}] \geq -1 \cdot L \wedge S[i_{\text{start}}] \leq L$ **do**
5: $i_{\text{start}} \leftarrow i_{\text{start}} + 1$
6: **end while**
7: **return** L, i_{start}
8: **end procedure**

Für das Zählen der Nulldurchläufe wird, beginnend mit dem Index i_{start}, über die gesamten Daten iteriert. In jeder Iteration wird geprüft, ob ein Nulldurchlauf stattgefunden hat. Dies ist dann der Fall, wenn, ausgehend vom letzten gefundenen Nulldurchlauf, der komplette Wertebereich zwischen den beiden Schwellwerten durchschritten wurde. Diese Überprüfung erfolgt mit Hilfe der Variable v_{last}. Ist deren Wert kleiner als L, so war der letzte Nulldurchlauf bei einem Wert kleiner als $-L$. Folglich ist der nächste Nulldurchlauf erreicht, wenn der aktuelle Wert L überschreitet. Umgekehrt gilt dies analog. Ist v_{last} größer als $-L$, so ist der nächste Nulldurchlauf erreicht, wenn der aktuelle Wert $-L$ unterschreitet. Wird ein Nulldurchlauf gefunden, so werden die zuvor initialisierten Variablen aktualisiert. Dabei wird die Zahl der Nulldurchläufe z_{count} inkrementiert sowie die Werte von i_{last} und v_{last} neu gesetzt. Ist der gefundene Nulldurchlauf der erste, wird

außerdem der Wert von i_{first} gesetzt. Wenn die for-Schleife abgeschlossen ist, werden die resultierenden Werte der Variablen z_{count}, i_{first} und i_{last} für die Berechnung der Frequenz verwendet.

Dazu wird zuerst die Rate der Nulldurchläufe *zcr* berechnet. Dabei wird, anstelle der Gesamtgröße der gegebenen Audiodaten N, der Abstand zwischen dem ersten und letzten gefundenen Nulldurchlauf verwendet ($i_{\text{last}} - i_{\text{first}}$). Bei diesem Ansatz befindet sich jeweils bei i_{first} und i_{last} ein Nulldurchlauf. Die Anzahl der gezählten Nulldurchläufe z_{count} ist daher um eins größer als die Anzahl der im Bereich enthaltenen Perioden. Dies muss deshalb bei der Berechnung von *zcr* explizit berücksichtigt werden. Basierend auf der Rate der Nulldurchläufe und der Samplerate f_S kann dann die Frequenz f berechnet und zurückgegeben werden.

Bei der Berechnung der Audiofrequenz kann dieses Vorgehen direkt so angewandt werden. Für die Schwebungserkennung ist hingegen noch eine Vorverarbeitung der Daten notwendig. Dort liegt das Problem vor, dass die Amplituden bzw. Fourierkoeffizienten nur positive Werte annehmen können. Folglich sind die Ausgangsdaten nicht um den Nullpunkt zentriert. Deshalb ist es notwendig, diese entsprechend entlang der y-Achse zu verschieben. Dafür wird der Durchschnitt $\bar{S}$ der Daten S berechnet:

$$\bar{S} = \frac{1}{N} \cdot \sum_{i=0}^{N-1} S[i]. \tag{A.12}$$

Dieser Wert wird an allen Positionen vom Signal S abgezogen. Auf die resultierenden Daten S' kann dann wieder die Frequenzerkennung angewendet werden:

$$S'[i] = S[i] - \bar{S}, \quad \forall i \in [0, N-1]. \tag{A.13}$$

Autokorrelation und quadratische Differenzfunktion

Der Pseudocode für die Frequenzerkennung per ACF ist in Algorithmus A.8 gegeben. An die entsprechende Funktion pitchDetectionByAcf() werden die folgenden Parameter übergeben:

- S: Audiodaten, die analysiert werden sollen.
- f_S: Samplerate der Audiodaten.
- W: Fenstergröße, die für die Berechnung verwendet werden soll.
- P_f: Faktor, der zur Erkennung der Maxima verwendet werden soll.

Algorithmus A.8 Algorithmus zur Frequenzerkennung per ACF.

Require: $S[0 \,..\, N-1] \in \mathbb{R}, \quad f_S \in \mathbb{N}, \quad W \in \mathbb{N}, \quad P_f \in [0,1]$

1: **procedure** PITCHDETECTIONBYACF($S[0 \,..\, N-1]$, f_S, W, P_f)
2: `// ACF berechnen`
3: $S_{\text{ACF}}[0 \,..\, N-W] \leftarrow \{0,\ ..,\ 0\}$
4: **for each** $i \in [0, N-W]$ **do**
5: **for each** $j \in [0, W-1]$ **do**
6: $S_{\text{ACF}}[i] \leftarrow S_{\text{ACF}}[i] + S[j] \cdot S[i+j]$
7: **end for**
8: **end for**

9: `// Frequenz berechnen`
10: $p_0, p_1 \leftarrow$ FINDACFPEAKS(S_{ACF}, P_f)
11: $f \leftarrow f_s \div (p_1 - p_0)$
12: **return** f
13: **end procedure**

Als Ergebnis wird die ermittelte Frequenz f zurückgegeben.

Im ersten Schritt wird das ACF-Ergebnis S_{ACF} der Größe $N - W + 1$ ausschließlich mit Nullwerten initialisiert. In der Hauptschleife wird über alle Positionen i iteriert, an denen das Fenster der Größe W vollständig über dem Signal S platziert werden kann. Die innere Schleife dient dazu, die Autokorrelation zwischen dem Teilbereich am Beginn des Signals (umfasst die Indizes 0 bis $W-1$) sowie dem aktuellen Fenster (umfasst die Indizes i bis $i + W - 1$) zu berechnen. Dabei wird über alle Indizes $j \in [0, W-1]$ iteriert und jeweils das Produkt der übereinanderliegenden Werte berechnet. Die Produkte werden aufsummiert und in das Ergebnisarray S_{ACF} geschrieben.

Für die Berechnung der gefundenen Frequenz müssen die Positionen der ersten beiden lokalen Maxima in S_{ACF} gefunden werden. Deren Abstand entspricht der Periodendauer der gesuchten Frequenz. Die Berechnung erfolgt mit Hilfe der Funktion FINDACFPEAKS() (siehe Algorithmus A.9). Für die Suche nach den lokalen Maxima wird ein Schwellwert v_{min} verwendet. Dieser befindet sich zwischen dem minimalen Wert MIN(S_{ACF}) und dem maximalen Wert MAX(S_{ACF}) des Ergebnisses. Der genaue Wert wird über den Faktor P_f festgelegt. Alle Bereiche von S_{ACF}, deren Werte größer als v_{min} sind, enthalten ein lokales Maximum.

Um diese Bereiche zu finden, wird über S_{ACF} iteriert. Wird dabei der Beginn eines solches Bereichs gefunden, so wird der aktuelle Index i in der lokalen Variable i_{start}

Algorithmus A.9 Algorithmus zum Finden der lokalen ACF-Maxima.

Require: $S_{\text{ACF}}[0 \,..\, N-W] \in \mathbb{R}, \quad P_f \in [0,1]$

1: **procedure** FINDACFPEAKS($S_{\text{ACF}}[0 \,..\, N-W]$, P_f)
2: $p \leftarrow$ CREATELIST()
3: $v_{\text{min}} \leftarrow \text{MIN}(S_{\text{ACF}}) + P_f \cdot (\text{MAX}(S_{\text{ACF}}) - \text{MIN}(S_{\text{ACF}}))$
4: $i_{\text{start}} \leftarrow -1$
5: **for each** $i \in [0, N-W]$ **do**
6: **if** $(S_{\text{ACF}}[i] \geq v_{\text{min}}) \wedge (i_{\text{start}} = -1)$ **then**
7: $i_{\text{start}} \leftarrow i$
8: **else if** $(S_{\text{ACF}}[i] < v_{\text{min}}) \wedge (i_{\text{start}} \neq -1)$ **then**
9: $i_{\text{max}} = i \in [i_{\text{start}}, i] \mid S_{\text{ACF}}[i] = \text{MAX}(S_{\text{ACF}}[i_{\text{start}} : i])$
10: APPEND(p, i_{max})
11: $i_{\text{start}} \leftarrow -1$
12: **end if**
13: **end for**
14: **return** $p[0]$, $p[1]$
15: **end procedure**

zwischengespeichert. Wird hingegen bei einem Index i das Ende eines solchen Abschnitts gefunden, wird das lokale Maximum genauer untersucht. Dafür wird der Index i_{max} gesucht, an dessen Position der maximale Wert des Bereichs $[i_{\text{start}}, i]$ zu finden ist. Dieser Wert entspricht der Position des gefundenen lokalen Maximums und wird zur Liste der Ergebnisse p hinzugefügt. Abschließend wird i_{start} wieder auf seinen initialen Wert zurückgesetzt, um das Finden weiterer Maxima zu ermöglichen. Nach Abschluss aller Iterationen werden die Indizes des ersten und zweiten Maximums zurückgegeben.

Zusammen mit der Samplerate f_S werden diese verwendet, um die resultierende Frequenz zu berechnen und als finales Ergebnis zurückzugeben.

Der Pseudocode für die Frequenzerkennung per SDF ist in Algorithmus A.10 gegeben. An die entsprechende Funktion PITCHDETECTIONBYSDF() werden die selben Parameter wie an PITCHDETECTIONBYACF() übergeben. Ihr Ablauf ist ebenfalls sehr ähnlich zur Berechnung per Autokorrelation.

Der einzige Unterschied ist, dass, anstelle der Summe der Produkte, die Summe der quadratischen Abweichungen zwischen den Werten der beiden Fenster berechnet wird. Das macht jedoch eine andere Handhabung der Ergebnisse notwendig. Bei der SDF liegen die höchsten Übereinstimmungen bei minimalen Werten vor. Die Funkti-

Algorithmus A.10 Algorithmus zur Frequenzerkennung per SDF.

Require: $S[0 \,..\, N-1] \in \mathbb{R}, \quad f_S \in \mathbb{N}, \quad W \in \mathbb{N}, \quad P_f \in [0,1]$

1: **procedure** pitchDetectionBySdf($S[0 \,..\, N-1]$, f_S, W, P_f)
2: `// SDF berechnen`
3: $S_{\text{SDF}}[0 \,..\, N-W] \leftarrow \{0,\ ..,\ 0\}$
4: **for each** $i \in [0, N-W]$ **do**
5: **for each** $j \in [0, W-1]$ **do**
6: $S_{\text{SDF}}[i] \leftarrow S_{\text{SDF}}[i] + (S[j] - S[i+j])^2$
7: **end for**
8: **end for**

9: `// Frequenz berechnen`
10: $p_0, p_1 \leftarrow$ findSdfPeaks(S_{SDF}, P_f)
11: $f \leftarrow f_s \div (p_1 - p_0)$
12: **return** f
13: **end procedure**

on findSdfPeaks() ist gegenüber findAcfPeaks() dementsprechend angepasst und untersucht das Ergebnisarray S_{SDF} nach lokalen Minima (siehe Algorithmus A.11).

Fourieranalyse

Der Pseudocode für die Frequenzerkennung per FFT ist in Algorithmus A.12 gegeben. An die entsprechende Funktion pitchDetectionByFft() werden die folgenden Parameter übergeben:

- S: Audiodaten, die analysiert werden sollen.
- f_S: Samplerate der Audiodaten.
- W: Fensterfunktion, die auf die Audiodaten angewendet werden soll. Unterstützt werden die Blackman- und Hanning-Fensterfunktion. Außerdem besteht die Möglichkeit, keine Fensterfunktion anzuwenden (Uniform-Fenster).

Als Ergebnis wird die ermittelte Frequenz f zurückgegeben.

Algorithmus A.11 Algorithmus zum Finden der lokalen SDF-Minima.

Require: $S_{\text{SDF}}[0 \,..\, N-W] \in \mathbb{R}, \quad P_f \in [0,1]$

procedure findSdfPeaks($S_{\text{SDF}}[0 \,..\, N-W]$, P_f)
  $p \leftarrow$ createList()
  $v_{\max} \leftarrow \text{min}(S_{\text{SDF}}) + P_f \cdot (\text{max}(S_{\text{SDF}}) - \text{min}(S_{\text{SDF}}))$
  $i_{\text{start}} \leftarrow -1$
  for each $i \in [0, N-W]$ **do**
    if $(S_{\text{SDF}}[i] \leq v_{\min}) \wedge (i_{\text{start}} = -1)$ **then**
      $i_{\text{start}} \leftarrow i$
    else if $(S_{\text{SDF}}[i] > v_{\min}) \wedge (i_{\text{start}} \neq -1)$ **then**
      $i_{\min} = i \in [i_{\text{start}}, i] \mid S_{\text{SDF}}[i] = \text{min}(S_{\text{SDF}}[i_{\text{start}} : i])$
      append($p, i_{\min}$)
      $i_{\text{start}} \leftarrow -1$
    end if
  end for
  return $p[0]$, $p[1]$
end procedure

Algorithmus A.12 Algorithmus zur Frequenzerkennung per FFT.

Require: $S[0 \,..\, N-1] \in \mathbb{R}, \quad f_S \in \mathbb{N}, \quad W \in \{\text{Uniform}, \text{Blackman}, \text{Hanning}\}$

procedure pitchDetectionByFft($S[0 \,..\, N-1]$, f_S, W)
  `// FFT berechnen`
  $S_W[0 \,..\, N-1] \leftarrow$ applyWindow(S, W)
  $S_{\text{FFT}}\left[0 \,..\, \frac{N}{2}\right] \leftarrow$ calcRealFft(S_W)

  `// Frequenz berechnen`
  $i_{\max} \leftarrow i \in \left[0, \frac{N}{2}\right] \mid S_{\text{FFT}}[i] = \text{max}(S_{\text{FFT}})$
  $i_{\max 2} \leftarrow$ interpolateFftIndex($S_{\text{FFT}}, i_{\max}$)
  $f \leftarrow i_{\max 2} \cdot (f_S \div N)$
  return f
end procedure

Im ersten Schritt wird die Fensterfunktion W auf die Ausgangsdaten S angewendet. Dafür wird die Funktion applyWindow() verwendet. Als Ergebnis gibt diese die resultierenden Daten S_W zurück. Auf diesen wird über die Funktion calcRealFft(S_W) das Amplitudenspektrum berechnet. Dabei wird der imaginäre Anteil der Fouriertransfor-

mation verworfen und nur der reelle Anteil zurückgegeben. Das resultierende Spektrum besteht aus $\frac{N}{2} + 1$ Werten, die die Frequenzanteile des Frequenzbereichs von 0 Hz bis $\frac{f_S}{2}$ enthalten. In diesem Spektrum wird der Index $i_{\max}$ des globalen Maximums gesucht.

Zur Erhöhung der Genauigkeit wird dieser interpoliert (siehe Kapitel A.1.2). Der resultierende Index $i_{\max 2}$ wird abschließend verwendet, um die finale Frequenz f zu berechnen und diese zurückzugeben.

Harmonic Product Spectrum

Der Pseudocode für die Frequenzerkennung per HPS ist in Algorithmus A.13 gegeben. An die entsprechende Funktion pitchDetectionByHps() werden die folgenden Parameter übergeben:

- S: Audiodaten, die analysiert werden sollen.
- f_S: Samplerate der Audiodaten.
- W: Fensterfunktion, die auf die Audiodaten angewendet werden soll. Unterstützt werden die Blackman- und Hanning-Fensterfunktion. Außerdem besteht die Möglichkeit, keine Fensterfunktion anzuwenden (Uniform-Fenster).
- H: Anzahl der Faktoren, die für die Berechnung des HPS verwendet werden sollen.

Als Ergebnis wird die ermittelte Frequenz f zurückgegeben.

Im ersten Schritt wird die Fensterfunktion W auf die Audiodaten S angewendet und das Amplitudenspektrum S_{FFT} berechnet. Dieses wird dann für die Berechnung des HPS verwendet.

Dazu wird, basierend auf der originalen Datengröße N und der Anzahl der zu verwendenden Faktoren H, dessen resultierende Größe N_{HPS} berechnet. Mit dieser Größe wird das Ergebnisarray S_{HPS} initialisiert. Für die Berechnung des HPS wird über alle Indizes iteriert, für die ein HPS-Wert berechnet werden kann. Die anderen Indizes werden nicht betrachtet, weil für diese nicht die benötigten Faktoren zur Verfügung stehen. In der inneren Schleife wird das Produkt der Fourierkoeffizienten aller zu betrachtenden Teiltöne, konkret dem Grundton und den ersten $H - 1$ Obertönen, berechnet und an die entsprechende Stelle im Ergebnis S_{HPS} geschrieben.

Aus diesem wird der Index $i_{\max}$ des maximalen HPS-Werts gesucht und per Interpolation verfeinert. Der resultierende Index $i_{\max 2}$ wird anschließend zur Berechnung der Frequenz f verwendet. Diese wird als Ergebnis zurückgegeben.

Algorithmus A.13 Algorithmus zur Frequenzerkennung per HPS.

Require: $S[0 \,..\, N-1] \in \mathbb{R}, \quad f_S \in \mathbb{N}, \quad W \in \{\text{Uniform}, \text{Blackman}, \text{Hanning}\}, \quad H \in \mathbb{N}$

1: **procedure** pitchDetectionByHps($S[0 \,..\, N-1]$, f_S, W, H)
2: `// FFT berechnen`
3: $S_W[0 \,..\, N-1] \leftarrow$ applyWindow(S, W)
4: $S_{\text{FFT}}\left[0 \,..\, \frac{N}{2}\right] \leftarrow$ calcRealFft(S_W)

5: `// HPS berechnen`
6: $N_{\text{HPS}} \leftarrow \lfloor \frac{N}{2 \cdot H} \rfloor$
7: $S_{\text{HPS}}[0 \,..\, N_{\text{HPS}} - 1] \leftarrow \{1, \,..,\, 1\}$
8: **for each** $i \in [0, N_{\text{HPS}} - 1]$ **do**
9: **for each** $j \in [1, H]$ **do**
10: $S_{\text{HPS}}[i] \leftarrow S_{\text{HPS}}[i] \cdot S_{\text{FFT}}[i \cdot j]$
11: **end for**
12: **end for**

13: `// Frequenz berechnen`
14: $i_{\max} \leftarrow i \in [0, N_{\text{HPS}} - 1] \mid S_{\text{HPS}}[i] = \text{max}(S_{\text{HPS}})$
15: $i_{\max 2} \leftarrow$ interpolateFftIndex($S_{\text{HPS}}, i_{\max}$)
16: $f \leftarrow i_{\max 2} \cdot (f_S \div N)$
17: **return** f
18: **end procedure**

Harmonic Spacing Analysis

Der Pseudocode für die Frequenzerkennung per HSA ist in Algorithmus A.14 gegeben. An die entsprechende Funktion pitchDetectionByHsa() werden die folgenden Parameter übergeben:

- S: Audiodaten, die analysiert werden sollen.
- f_S: Samplerate der Audiodaten.
- W: Fensterfunktion, die auf die Audiodaten angewendet werden soll. Unterstützt werden die Blackman- und Hanning-Fensterfunktion. Außerdem besteht die Möglichkeit, keine Fensterfunktion anzuwenden (Uniform-Fenster).
- P_f: Faktor, der zur Erkennung der Maxima verwendet werden soll.

Algorithmus A.14 Algorithmus zur Frequenzerkennung per HSA.

Require: $S[0 \,..\, N-1] \in \mathbb{R}, \quad f_S \in \mathbb{N}, \quad W \in \{\text{Uniform}, \text{Blackman}, \text{Hanning}\}, \quad P_f \in \mathbb{R}$

1: **procedure** pitchDetectionByHsa($S[0 \,..\, N-1]$, f_S, W, P_f)
2: `// FFT berechnen`
3: $S_W[0 \,..\, N-1] \leftarrow$ applyWindow(S, W)
4: $S_{\text{FFT}}\left[0 \,..\, \frac{N}{2}\right] \leftarrow$ calcRealFft(S_W)

5: `// FFT-Maxima finden und Abstände berechnen`
6: $p \leftarrow$ findFftPeaks(S_{FFT}, P_f)
7: $d \leftarrow$ createList()
8: **for each** $i \in [0, \text{size}(p) - 2]$ **do**
9: append($d, p[i+1] - p[i]$)
10: **end for**
11: sort(d)

12: `// Frequenz berechnen`
13: $i_{\max} \leftarrow$ executeClustering(d, 0,015, true)
14: $f \leftarrow i_{\max} \cdot (f_S \div N)$
15: **return** f
16: **end procedure**

Als Ergebnis wird die ermittelte Frequenz f zurückgegeben.

Im ersten Schritt wird die Fensterfunktion W auf die Audiodaten S angewendet. Vom resultierendem Signal S_W wird das Amplitudenspektrum S_{FFT} berechnet. Anschließend werden die lokalen Maxima in diesem gesucht. Dies erfolgt mithilfe der Funktion findFftPeaks() (siehe Algorithmus A.15).

Der Algorithmus A.15 funktioniert ähnlich wie die Algorithmen A.9 und A.11 bei der Frequenzerkennung per ACF bzw. SDF. Zuerst wird ein Schwellwert $v_{\min}$ berechnet. Dafür wird der Durchschnitt aller FFT-Werte gebildet und mit dem Faktor P_f multipliziert. Alle Bereiche mit einem Wert größer als $v_{\min}$ entsprechen einem lokalen Maximum. Durch Iteration über alle Indizes des Amplitudenspektrums S_{FFT} werden die entsprechenden Bereiche und Maxima gesucht. Deren Indizes werden jeweils zur Ergebnisliste p hinzugefügt. Zuvor werden diese, zur Erhöhung der Genauigkeit, interpoliert. Abschließend wird die erstellte Liste p zurückgegeben.

Für die weitere Berechnung werden die Abstände zwischen allen benachbarten Maxima berechnet. Dazu wird über die Liste p iteriert und die jeweiligen Differenzen zur Liste d

Algorithmus A.15 Algorithmus zum Finden der FFT-Maxima.

Require: $S\left[0 \,..\, \frac{N}{2}\right] \in \mathbb{R}, \quad P_f \in \mathbb{R}$

procedure findFftPeaks($S_{\text{FFT}}\left[0 \,..\, \frac{N}{2}\right]$, P_f)
  $p \leftarrow$ createList()
  $v_{\min} \leftarrow P_f \cdot \frac{2}{N} \cdot \sum_{i=0}^{\frac{N}{2}} S_{\text{FFT}}[i]$
  $i_{\text{start}} \leftarrow -1$
  for each $i \in \left[0, \frac{N}{2}\right]$ **do**
    if $(S_{\text{FFT}}[i] \geq v_{\min}) \wedge (i_{\text{start}} = -1)$ **then**
      $i_{\text{start}} \leftarrow i$
    else if $(S_{\text{FFT}}[i] < v_{\min}) \wedge (i_{\text{start}} \neq -1)$ **then**
      $i_{\max} = i \in [i_{\text{start}}, i] \mid S_{\text{FFT}}[i] = \max(S_{\text{FFT}}[i_{\text{start}} : i])$
      $i_{\max 2} \leftarrow$ interpolateFftIndex($S_{\text{FFT}}, i_{\max}$)
      append($p, i_{\max 2}$)
      $i_{\text{start}} \leftarrow -1$
    end if
  end for
  return p
end procedure

hinzugefügt. Diese wird dann aufsteigend sortiert und für die Durchführung des Clusterings verwendet. Dabei werden auch Cluster, die Vielfache voneinander sind, zusammengefasst (angegeben durch den Wert true für den dritten Parameter). Als maximale relative Differenz zwischen Clustern wird 0,015 gewählt, so dass nur Cluster zusammengefasst werden deren Wertunterschied maximal 1,5 % bzw. 101,5 %, 201,5 % usw. zueinander beträgt. Das resultierende Ergebnis des Clusterings $i_{\max}$ wird für die Berechnung der Frequenz f verwendet, die abschließend zurückgegeben wird.

Cepstrum

Der Pseudocode für die Frequenzerkennung per Cepstrum ist in Algorithmus A.16 gegeben. An die entsprechende Funktion pitchDetectionByCepstrum() werden die folgenden Parameter übergeben:

- S: Audiodaten, die analysiert werden sollen.
- f_S: Samplerate der Audiodaten.

- W: Fensterfunktion, die auf die Audiodaten angewendet werden soll. Unterstützt werden die Blackman- und Hanning-Fensterfunktion. Außerdem besteht die Möglichkeit, keine Fensterfunktion anzuwenden (Uniform-Fenster).
- I_f: Anteil der Cepstrum-Daten, die bei der Berechnung des Ergebnisses verworfen werden sollen.

Als Ergebnis wird die ermittelte Frequenz f zurückgegeben.

Algorithmus A.16 Algorithmus zur Frequenzerkennung per Cepstrum.

Require: $S[0 \,..\, N-1] \in \mathbb{R}$, $f_S \in \mathbb{N}$, $W \in \{\text{Uniform}, \text{Blackman}, \text{Hanning}\}$, $I_f \in [0, 1]$

1: **procedure** PITCHDETECTIONBYCEPSTRUM($S[0 \,..\, N-1]$, f_S, W, I_f)
2: `// Cepstrum berechnen`
3: $S_W[0 \,..\, N-1] \leftarrow$ APPLYWINDOW(S, W)
4: $S_{\text{FFT}}\left[0 \,..\, \frac{N}{2}\right] \leftarrow$ CALCREALFFT(S_W)
5: $S_{\text{FFT,ln}}\left[0 \,..\, \frac{N}{2}\right] \leftarrow \ln(S_{\text{FFT}})$
6: $S_{\text{Cep}}\left[0 \,..\, \frac{N}{4}+1\right] \leftarrow$ CALCREALFFT($S_{\text{FFT,ln}}$)

7: `// Frequenz berechnen`
8: $i_{\text{Cep,low}} \leftarrow \left\lfloor I_f \cdot (\frac{N}{4}+1) \right\rfloor$
9: $i_{\text{Cep,high}} \leftarrow \frac{N}{4}+1$
10: $i_{\max} \leftarrow i \in [i_{\text{Cep,low}}, i_{\text{Cep,high}}] \mid S_{\text{Cep}}[i] = \text{MAX}(S_{\text{Cep}}\,[i_{\text{Cep,low}} : i_{\text{Cep,high}}])$
11: $i_{\max 2} \leftarrow$ INTERPOLATEFFTINDEX($S_{\text{Cep}}, i_{\max}$)
12: $f \leftarrow (1 \div i_{\max 2}) \cdot ((\frac{N}{4}+1) \div N) \cdot f_S$
13: **return** f
14: **end procedure**

Im ersten Schritt wird das Cepstrum der Audiodaten S berechnet. Dazu wird zuerst die Fensterfunktion W angewendet und das Amplitudenspektrum S_{FFT} berechnet. Auf dieses wird dann elementweise der natürliche Logarithmus angewendet. Durch erneute Anwendung der Fouriertransformation auf das resultierende Signal $S_{\text{FFT,ln}}$ kann dann das Cepstrum S_{Cep} berechnet werden.

Für die Berechnung der gefundenen Frequenz f wird das Maximum $i_{\max}$ des Cepstrums gesucht. Dabei wird der untere Teilbereich des Cepstrums nicht mit betrachtet. Grund dafür ist, dass sich dort ein Maximum für die tieffrequenten Anteile (resultierend aus der Hüllkurve des Amplitudenspektrums) befindet. Der konkrete Umfang des zu ignorierenden Anteils wird über den Parameter I_f spezifiziert. Das somit gefundene Maximum wird

ebenfalls, zur Erhöhung der Genauigkeit, interpoliert. Der resultierende Index i_{max2} wird dann zur Berechnung der Frequenz f verwendet. Diese wird abschließend zurückgegeben.

A.2 Parametrierung der evaluierten Algorithmen

Dieses Kapitel enthält eine tabellarische Auflistungen aller für die Evaluation verwendeten Parametrierungen der Frequenz- und Schwebungserkennung.

A.2.1 Algorithmen für die Frequenzerkennung

Die folgende Tabelle A.1 enthält die Parametrierungen, die bei der Evaluation der Frequenzerkennung, sowohl mit den additiven als auch den GrandOrgue-basierten Testdaten, verwendet wurden.

Verfahren	**Verwendete Parameter**
ACF	• $P_f = 0{,}95$
SDF	• $P_f = 0{,}02$
ZCR	–
Cepstrum	• $W =$ Blackman • $I_f = 0{,}012$
FFT	• $W =$ Uniform
HPS	• $W =$ Hanning • $H = 13$
HSA	• $W =$ Hanning • $P_f = 8$

Tabelle A.1: Parametrierungen der Frequenzerkennungsverfahren.

A.2.2 Algorithen für die Schwebungserkennung

Analyse der Amplituden

Die folgende Tabelle A.2 enthält die Parameterierungen, die bei der Evaluation der Schwebungserkennung per Analyse der Amplituden für die additiven Testdaten verwendet wurden.

Verfahren	**Verwendete Parameter**
ACF	• $V_{\text{nobeat}} = 0{,}95$ • $P_f = 0{,}82$
SDF	• $V_{\text{nobeat}} = 0{,}95$ • $P_f = 0{,}21$
ZCR	• $V_{\text{nobeat}} = 0{,}95$
Cepstrum	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $I_f = 0{,}100$
FFT	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $I_f = 0{,}024$
HPS	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Uniform • $H = 3$
HSA	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $P_f = 0{,}9$

Tabelle A.2: Parametrierungen der Schwebungserkennungsverfahren per Analyse der Amplituden mit additiven Testdaten.

Analyse der Hüllkurve

Die folgende Tabelle A.3 enthält die Parametrierungen, die bei der Evaluation der Schwebungserkennung per Analyse der Hüllkurve für die additiven Testdaten verwendet wurden.

Verfahren	**Verwendete Parameter**
ACF	• $V_{\text{nobeat}} = 0{,}95$ • $P_f = 0{,}82$
SDF	• $V_{\text{nobeat}} = 0{,}95$ • $P_f = 0{,}21$
ZCR	• $V_{\text{nobeat}} = 0{,}95$
Cepstrum	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $I_f = 0{,}100$
FFT	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $I_f = 0{,}024$
HPS	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Uniform • $H = 3$
HSA	• $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $P_f = 0{,}9$

Tabelle A.3: Parametrierungen der Schwebungserkennungsverfahren per Analyse der Hüllkurve mit additiven Testdaten.

Analyse der Fourierkoeffizienten

Die folgende Tabelle A.4 enthält die Parametrierungen, die bei der Evaluation der Schwebungserkennung per Analyse der Fourierkoeffizienten für die additiven Testdaten verwendet wurden. Die Parametrierungen für die GrandOrgue-basierten Testdaten sind in der Tabelle A.5 gegeben.

Verfahren	**Verwendete Parameter**
ACF	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}95$ • $P_f = 0{,}70$
SDF	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}95$ • $P_f = 0{,}37$
ZCR	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}95$
Cepstrum	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $I_f = 0{,}060$
FFT	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $I_f = 0{,}008$
HPS	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $H = 2$
HSA	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}95$ • $W =$ Hanning • $P_f = 7$

Tabelle A.4: Parametrierungen der Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit additiven Testdaten.

Verfahren	**Verwendete Parameter**
ACF	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}80$ • $P_f = 0{,}70$
SDF	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}85$ • $P_f = 0{,}37$
ZCR	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}85$
Cepstrum	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}80$ • $W =$ Hanning • $I_f = 0{,}060$
FFT	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}85$ • $W =$ Hanning • $I_f = 0{,}008$
HPS	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}80$ • $W =$ Hanning • $H = 2$
HSA	• $n_T = 4$ • $V_{\text{nobeat}} = 0{,}85$ • $W =$ Hanning • $P_f = 7$

Tabelle A.5: Parametrierungen der Schwebungserkennungsverfahren per Analyse der Fourierkoeffizienten mit GrandOrgue-basierten Testdaten.

Selbstständigkeitserklärung

Name:	Junghanns
Vorname:	Nico
geb. am:	25.12.1997
Matr.-Nr.:	435716

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Chemnitz, 1. November 2022

Nico Junghanns