Umsetzbarkeit der Trennung von Percussive und Melodic Frequenzen in einer bereits komprimierten Wav Datei

Projektteil der Belegung einer Wahlspezialisierung
im Studiengang Informatik
an der Fakultät für Informatik und Ingenieurwissenschaften
der Technischen Hochschule Köln

vorgelegt von:	Lukas Fey, Nicolas Friedmann
Matrikel-Nr.:	123 456 789, 111 55 463
Adresse:	Auf der Platte. 1 51643 Gummersbach vorname.nachname@smail.th-koeln.de, nicolas-friedmann@gmx.de
eingereicht bei:	Prof. Dr. Lutz Köhler
Zweitgutachter*in:	Prof. Dr. Vorname Nachname

Ort, TT.MM.JJJJ

Kurzfassung/*Abstract*

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Musik ist zu einem Teil des täglichen Labens vieler Menschen geworden. Durch den Konsum und den wirtschaftlichen Ertrag wird an der Produktion und Analyse von Musik geforscht (Quelle?).

Ein Teil der Forschung bezieht sich auf die Trennung einer Wellenfunktion (einheitlichen Fachbegriff finden und verwenden?) in die unterschiedlichen Funktionen der Frequenzen. Zuvor beinhaltet die Wellenfunktion eine oder mehrere Sinus- und Kosinusfunktionen, die eine neue Funktion bilden. Dies wird sowohl auf audiovisuelle als auch auf visuelle Funktionen angewendet. Beispielsweise kann die Funktion mehrere Instrumente beinhalten, die anhand der Funktion nicht identifizierbar sind. Einer der Algorithmen zur Trennung von Signalen heißt Fourier-Transformation und wird in diesem Projekt behandelt.

Unter anderem werden Musikinstrumente in Liedern getrennt und einzeln angehört oder wiederverwendet. In diesem Projekt werden zum Einstieg lediglich perkussive aus akkustischen Instrumenten getrennt. Akkustische Instrumente sind Instrumente die einen Ton erzeugen indem Menschen Kraft auf sie ausüben. Perkussive Instrumente werden geschlagen, geschütttelt oder geschabt und sind eine spezielle Form der akkustischen Instrumente. Die übrigen Instrumente sind harmonisch und ebenfalls akkustische Instrumente.

(https://www.metromusicmakers.com/2020/08/the-difference-between-acoustic-electric-and-digital-instruments/)

(Frequenzen und Wellenfunktion woanders erklären?)

1.1 Relevanz

Es gibt unterschiedliche Kontexte, in denen die Trennung von Audiosignalen zum Einsatz kommt. Häufig ist eine Eingabefunktion als Summe aller Signale schwierig zu analysieren oder weiterzuverarbeiten. Beispielsweise, wenn einzelne Instrumente im Nachhinein bearbeitet werden sollen.

In Zeiten von zunehmend digital produzierter Musik und Verbreitung von Informatik stellt sich die Frage, wie Musiksignale digital aufgebaut sind und wie man mit ihnen

arbeiten kann. Im Kontext dieser Arbeit wird die Fourier-Transformation verwendet, um eine Tondatei in Gruppen von harmonischen und perkussiven Musikinstrumenten zu zerlegen. Dies kann mit weiterer Modifikation verwendet werden, um bestimmte Instrumente zum üben zu trennen oder Hintergrundgeräusche auszublenden.

(Etwas zu kurz?)

1.2 Vorgehen

Ohne jegliches Vorwissen, wie die Signale einer Musikdatei gespeichert und von einem Computer interpretiert oder getrennt werden, wird das Projekt durchgeführt. Die Erkenntnisse und notwendiges Hintergrundwissen werden dokumentiert und erklären die Logik und Zusammenhänge der Fourier-Transformation. Durch die Dokumentation wird anderen Leser:innen ein erstes Verständnis von WAV-Dateien vermittelt auf dem im Anschluss aufgebaut werden kann.

(WAV-Dateien sehr spezifisch: Den Leser:innen wird auch die Funktionalität der FFT und Code etc. vermittelt?)

Das Projekt dokumentiert anschließend die Implementierung eines Algorithmus zur Trennung der harmonischen und perkussiven Tongruppen einer WAV-Datei mittels Fourier-Transformation, sowie das schreiben auf zwei getrennte WAV-Dateien mittels Inverse Fourier Transform (siehe: Kapitel 4). Dabei wird zuvor behandelte Logik und Hintergrundwissen referenziert und in der Praxis erklärt.

2 Stand der Wissenschaft

Die Trennung von Musikinstrumenten ist ein praktischer Anwendungsfall zur Separation von Signalen. Dabei liegen in einer Aufnahme mehrere Signale in Form von einer Funktion vor. In dieser Funktion sind die einzelnen Signale schwierig zu identifizieren.

Bei Vorgehen zur Trennung von Signalen wird die Funktion in eine Abhängigkeit zur Frequenz umgeformt. Anhand der Frequenz können unterschiedliche Signale identifiziert werden. Beispielsweise werden die Frequenzen unterschiedlichen Instrumenten zugeordnet.

Inzwischen wurden mehrere Möglichkeiten zur Transformation einer Funktion in Abhängigkeit zur Zeit in die Abhängigkeit zur Frequenz entdeckt. Diese unterscheiden sich in der Darstellung der Funktion und der Genauigkeit des Ergebnisses.

In diesem Projekt wird die Fourier-Transformation angewendet, die verbreitet und sehr effektiv ist (Quelle?). Es wird überprüft, ob die Implementierung und Anwendung ohne Vorkenntnisse und anspruchsvolle Hardware umsetzbar ist.

Eine weitere Methode ist die Wavelet-Transformation. Diese wird in Abschnitt 3.4.2 erläutert, um die Entscheidung für die Fourier-Transformation zu begründen.

3 Hauptteil

Um ein grundlegendes Verständnis für die Analyse von Dateiformaten zu entwickeln wird einem klaren Schema gefolgt. Mit diesem Schema wird sich an der chronologischen Reihenfolge von der Entstehung eines Tons bis zur Verarbeitung im Code orientiert.

Jeder Ton wird durch eine Vibration erzeugt. Diese Vibration kann durch unterschiedliche Gegenstände erzeugt werden. Dazu zählen Becken eines Schlagzeugs, Saiten eines Basses oder die Stimmbänder (?).

- falsches Kapitel („eher Was ist ein Ton?“) - Unterschiedliche Töne akkustisch, harmonisch und perkussiv zuordnen

1. Was ist ein Ton
2. Formen der Musikdarstellung
3. (Darstellung in der Mathematik)
4. Vorstellung Fourier-Transformation
5. Alternative Methode zur Fourier-Transformation
6. Trennung von Instrumenten im Code
7. Vorgehen im Projekt (evtl früher)
8. Wie kann man auf dem Stand aufbauen? (/Wie könnte es weitergehen?)
9. Fazit
    a) Aufwand
    b) Ertrag (z.B. erworbenes Wissen)

Um einen Algorithmus zu implementieren, ist es hilfreich die unterschieden Musikdarstellungen kennenzulernen. Für eine effektive Fourier-Transformation gibt es Vor. Manche Voraussetzungen für eine effektive Fourier-Transformation werden nur von wenigen Musikdarstellungen erfüllt.

Außerdem ist es relevant ein Verständnis für die Fourier-Transformation und andere Methoden zu entwickeln, um nachzuvollziehen warum die Fourier-Transformation geeignet ist. Zudem hilft ein Verständnis bei der Implementierung des Codes.

3.1 Was ist ein Ton?

Töne entstehen durch die Vibration eines Gegenstandes. Diese erzeugen Schallwellen. Der Luftdruck einer Schallwelle wird graphisch als eine Sinus- oder Cosinusfunktion dargestellt.

(Spektogramm?)

- Was sind Frequenzen Lautstärke etc. (Stauchung/ Streckung..., Verlauf) - Frequenz bei 4 Perioden/ Maxima in einer Sekunde -> 4Hz? - "The sinusoid can be considered the prototype of an acoustic realization of a musical note. Sometimes the sound resulting from a sinusoid is called a harmonic sound or pure tone"

3.2 Formen der Musikdarstellungen

(Einleitung und Überschriften?)

Musik kann unterschiedlich dargestellt werden. Je nach Bedarf, wird eine andere Form der Musikdarstellung benötigt. Eine Form der visuellen Musikdarstellung sind Musiknoten. Es ist eine formale Sprache, die vorgibt wie ein Musikstück gespielt wird [10].

- MIDI nutzt Notendarstllung

Eine weitere Form der Darstellung, ist die Audiodarstellung. In diesen Darstellungen werden alle Informationen der Töne als Audiosignale digital gespeichert und geteilt. Es werden die Schallwellen eines Tons (siehe ...) aufgenommen und digital als eine Wellenfunktion gespeichert, die an der Schallwelle orientiert ist. Dazu gehören auch das Timing, die Intensität, die Lautstärke, die Länge des Tons und vieles mehr. Es werden nicht die einzelnen Töne und Noten gespeichert, sondern die Frequenzen während der Aufnahme in Abhängigkeit zur Zeit. Die Audiodatei kann auch Nebengeräusche oder weitere Instrumente beihalten.

3.2.1 Einordnung in den Projektkontext

Die digitale Darstellung macht es schwieriger unterschiedliche Audiosignale zu trennen und die ursprünglichen Töne wieder herzustellen. Die verbreitetste Form der Audiodarstellung ist MP3 (Quelle?). In diesem Projekt werden jedoch Wav(einheitlich?)-Dateien behandelt. Die Unterschiede, sowie Vor- und Nachteile der Audiodarstellungen werden in Abschnitt 3.3.3 behandelt [11].

- Was ist eine Note audio-/ visuell

3.3 Aufbau einer Audiodatei/ Music Representation

Für die Aufnahme von Audiosignalen werden analoge Signale in eine digitale Form von Schall umgewandelt. Die analoge Signaldarstellung verwendet kontinuierliche und konstante Spannungsschwankungen, die den verursachten Luftdruckschwankungen entsprechen [5]. In der digitalen Form werden die Spannungsschwankungen als Bitstreams gespeichert und je nach Audio-Format komprimiert.

- Entstehung der Audiosignale: Analog/ Digital

3.3.1 Durchführung der Komprimierung

Die Komprimierung von Audiodateien wird meistens verwendet, um mehr Musik auf einem Datenträger speichern zu können [3]. Um eine Aufnahme zu komprimieren wird diese in eine Sampling Rate und eine Quantisierungsschrittweite reduziert.

Sampling Rate

Die Sampling Rate (/Sampling-Rate?) definiert die Anzahl von gespeicherten Signalen pro Sekunde. Dies reduziert die benötigten gespeicherten Daten von einer durchgängigen Aufnahme auf mehrere Blöcke. Beispielsweise hat eine CD Aufnahme eine Sampling Rate von 44.100 Samples pro Sekunde (kurz: 44.1 kHz), das heißt es werden 44.100 Soundsignale pro Sekunde gespeichert, dessen Übergänge kaum wahrnehmbar sind, jedoch bereits zu einem deutlichen Reduzierung des Speicherbedarfs führen.

(Bild einfügen?)

Quantisierungsschrittweite

Die Quantisierungsschrittweite beschreibt die Auflösung, mit der jedes Sample komprimiert wird. Die möglichen Werte eines Samples sind kontinuierlich und könnten theoretisch unendlich viele Dezimalstellen haben. Durch die Quantisierungsschrittweite werden diese Werte jedoch auf eine festgelegte Anzahl möglicher diskreter Werte reduziert, wodurch der Unterschied zwischen den Werten bestimmt wird. Bei CDs wird beispielsweise eine 16-Bit-Codierung gewählt, die 65.536 mögliche Werte definiert.

(Bild einfügen?)

3.3.2 Waveform Audio File Format

Das Waveform Audio File Format (kurz: WAV) speichert Audioaufnahmen unkomprimiert [13]. Der Begriff leitet sich „vom englischen Wort ‚wave'“ für Schallwelle ab [4]. Für die in diesem Projekt behandelte (Schnelle?) Fourier-Transformation sind insbesondere die „Abtastrate fs des Messsystems“ und die Quantisierungsschrittweite (siehe Kapitel...?) entscheidend [12]. Aufgrund der unkomprimierten Speicherung (und der damit verbundenen hohen Abtastrate und Quantisierungsschrittweite) lässt sich eine klare Trennung von Tonspuren ermöglichen (Referenz?).

3.3.3 Andere Formen von Audio-Dateien

- MP3
- WMA
- AAC
- OGG
- FLAC
- RM

[8]

3.4 Trennung einer Tonspur in verschiedene Instrumente

Eine Aufnahme speichert das aufgenommene Signal in Abhängigkeit zur Zeit. Jedes eingehende Signal verfügt über Schwingungen in Form von einer Amplitude. Bei einer Aufnahme vermischen sich die verschiedenen Signale zu einer gemeinsamen Amplitude und sind daher schwierig voneinander zu unterscheiden, beispielsweise bei Hintergrundgeräuschen oder der gleichzeitigen Aufnahme mehrerer Musikinstrumente.

Die Trennung von Instrumenten innerhalb einer Tonspur ist ein Thema, das in Forschung und Bildung intensiv behandelt wird (Quelle?). Die am häufigsten verwendete Methode ist die Fourier-Transformation (Quelle?), die es ermöglicht, ein Signal in seine Frequenzkomponenten zu zerlegen.(Gibt es andere Vorgehensweisen?).

Neben der Fourier-Transformation werden auch andere Ansätze wie die Wavelet-Transformation, die Short-Time Fourier Transform (STFT), sowie statistische Verfahren wie die Blind Source Separation (BSS) und die Independent Component Analysis

(ICA) eingesetzt. Darüber hinaus finden moderne Verfahren des maschinellen Lernens, insbesondere neuronale Netzwerke, zunehmend Anwendung in der Trennung von Audiosignalen.

3.4.1 Fourier-Transformation

Die Fourier-Transformation ist ein Algorithmus der die Darstellung einer Tondatei verändert. Ursprünglich liegt die Audiospur mit einer Kombination aus unterschiedlichen Frequenzen in Abhängigkeit zur Zeit vor. In dieser Darstellung sind die unterschiedlichen Signale schwierig zu trennen und werden von der Fourier-Transformation transformiert.

Entwicklung der Fourier-Transformation

Die Fourier-Transformation ist eine Verallgemeinerung der Fourierreihen. Diese Reihen können stetige oder stückweise stetige Funktionen in eine Summe von Sinus- und Kosinusfunktionen zerlegen. Bereits im 18. Jahrhundert wurden Fourierreihen für spezifische Funktionen entdeckt. 1822 stellte Joseph Fourier die Hypothese auf, dass sich jede Funktion als Summe solcher Reihen darstellen lässt. Erst im 20. Jahrhundert wurden Fourierreihen auch für andere stetige oder stückweise stetige Funktionen formal bewiesen. Dank der Vollständigkeiten der Funktionenreihe lässt sich die Fourier-Transformation auf eine Vielzahl von Funktionen anwenden, einschließlich periodischer und nicht-periodischer Funktionen, und erhielt ihren Namen zu Ehren von Fourier.

Durchführung der Transformation

Die Fourier-Transformation ist ein mathematisches Verfahren, bei dem ein Signal aus dem Zeitbereich in den Frequenzbereich transformiert wird. Die Transformation ermöglicht es, beliebige periodische und stückweise stetige Funktionen als Summe von Sinus- und Kosinuswellen unterschiedlicher Frequenzen darzustellen.

- Vergleichsschwingungen

In der neuen Darstellung werden die Frequenzen der Funktion unabhängig von der zeitlichen Komponente wiedergegeben. Unterschiedliche Frequenzen können unterschiedlichen Signalen zugeordnet werden. Die frequentielle Darstellung gibt an welche Signale in welchen Frequenzen Teil der Funktion sind, allerdings nicht wann. Daher wird die Darstellung wieder umgeformt in die zeitliche Abhängigkeit.

Das Signal oder die Signale, die von der Tonspur getrennt werden, können in der frequentiellen Darstellung identifiziert werden. Anschließend werden diese von den übrigen Signalen getrennt und zurück in die ursprüngliche Darstellung transformiert (Inverse Fourier Transform?). Damit erhält man eine neue Tondatei, die ausschließlich aus den benötigten Signalen besteht.

Anschließend werden die Tonspuren je nach Frequenz extrahiert und der Vorgang wieder rückgängig gemacht. Somit liegen diese als WAV-Datei vor und können weiterverwendet werden.

(Bild?)

Parameter der Transformation

- Probleme: - Rechenaufwand/ Dauer

In (...?) wurden die relevanten Parameter wie Samplingrate und Quantisierungsschrittweite behandelt, die für die Durchführbarkeit der Transformation entscheidend sind. Zudem können verschiedene Komponenten der Transformation konfiguriert bzw. parametrisiert werden.

Blocklänge (/engl.?)

Ein wesentlicher Parameter für die Durchführung der Fourier-Transformation ist die Blocklänge (kurz: BL), welche die Anzahl der Samples festlegt, die in der Fourier-Transformation analysiert werden.

Umso länger die BL, desto präziser ist das Ergebnis der Transformation. Dies erhöht jedoch auch den Rechenaufwand und die Dauer der Transformation. In einigen Anwendungsszenarien erfordert dies eine Abwägung zwischen Verarbeitungsdauer und Genauigkeit, beispielsweise in einer App zum Stimmen von Musikinstrumenten oder zum Erkennen der gespielten Töne (Quelle?).

Durchführung am Beispiel einer Musiknote

In diesem Beispiel (Müller referenzieren?) wird eine Note auf einem Piano gespielt und durch die Transformation in eine frequentielle Darstellung der gespielte Ton erkannt.

Die Aufnahme des Tons ist in Abbildung 3.1(a) zu erkennen. Für die Transformation wird ein Ausschnitt von 10ms verwendet. Das entspricht der Blocklänge und reduziert den Rechenaufwand (referenzieren?).

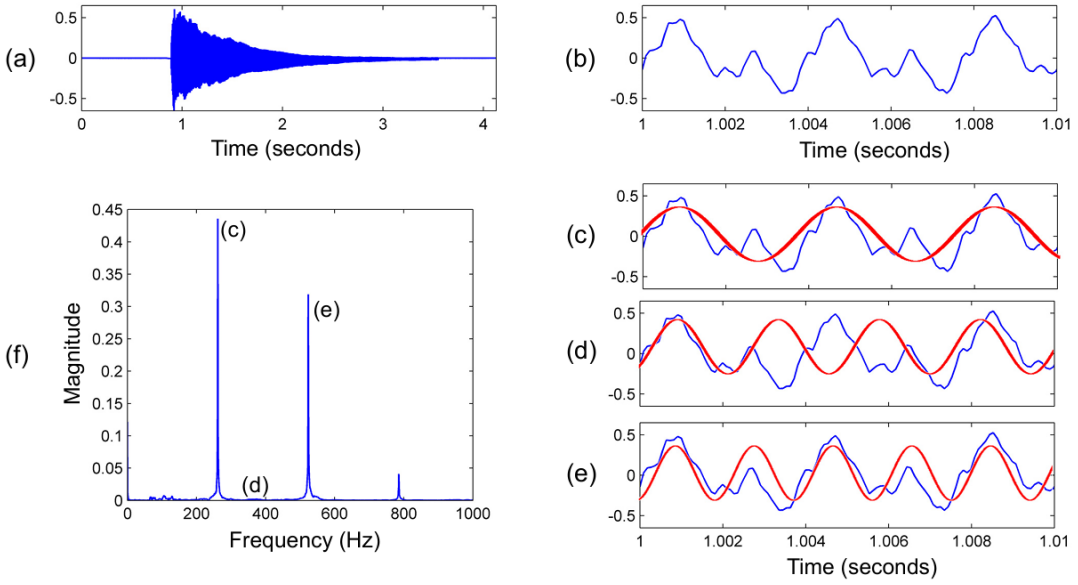

Abbildung 3.1: Note C4 in unterschiedlichen Darstellungen

Anschließend werden unterschiedliche Vergleichsfunktion für die jeweiligen Frequenzen mit dem Ausschnitt der Tonspur verglichen. Die Ähnlichkeiten der jeweiligen Frequenzen werden in (f) wiedergegeben.

In Abbildung 3.1(c) ist die Übereinstimmung für die Frequenz w = 262 Hz besonders hoch. Daraus folgt in (f) bei ungefähr 262 der höchste Wert. Die Höhe des Wertes wird in der Variable dw angegeben.

Die Frequenz 262 entspricht der Note C4. Darüberhinaus wird bei einer Frequenz von 523 (siehe Abbildung 3.1(e)) eine hohe Übereinstimmung erkannt. Dies entspricht ungefähr der Frequenz des zweiten Teils der Note C4 (zweiter Teil?).

Nachteile

Die Fourier-Transformation ermöglicht es je nach Bedarf zwischen der zeitlichen oder der sequentiellen Darstellung zu wechseln. Allerdings ist bei der Fourier-Transformation der Wechsel zwischen den Darstellung notwendig und es wird entweder die zeitliche oder die frequentielle Komponente ignoriert. Bei der Anwendung der Fourier-Transformation gibt es keine Darstellung die beide Komponenten kombiniert.

- Darstellung (nie Zeit und Frequenz)

(Noch unklar wie unterschiedliche Instrumente vorliegen, aber analoge Instrumente sollten in Sinusfunktion vorliegen und erkennbar sein) (Lukas: Alle Instrumente bzw

Töne der Instrumente liegen als Sinuswellen vor)

3.4.2 Wavelet-Transformation

Die Wavelet-Transformation ist ein Vorgehen, das eine Funktion der zeitlichen Darstellung in eine 3(-?)dimensionale Darstellung in Abhhängigkeit von Zeit und Frequenz transformiert. Dafür werden Wellenfunktionen mit der ursprünglichen Funktion auf Übereinstimmungen verglichen. Es gibt unterschiedliche Wellenfunktionen und sie werden Wavelets genannt. Der Name ist aus dem französichen und wird in kleine Welle oder Wellchen übersetzt.

Im Gegensatz zu der ursprünglichen Funktion sind die Flächen der Wavelets endlich und daher der Name (finite energy). Eine weitere Bedingung für Wavelets ist, dass das Integral null ergibt. Also die Fläche über und unter der X-Achse gleichgroß ist (Admissibility condition). Jedes Wavelet wird um die Parameter m und b ergänzt.

m: Bestimmt die Frequenz der Wavelet

b: Bestimmt den Zeitpunkt der Wavelet

Zudem hat das Wavelet einen realen und einen imaginären Teil. Das bedeutet, dass durch die Ergänzung der imaginären Zahl die Wavelet um eine Achse ergänzt wird und 3(-?)dimensional wird. Sowohl der reale als auch der imaginäre Teil der Wavelet werden mit der Funktionen verglichen und durch die Multiplikation (Begriff?) erhält man die Ähnlichkeit der Funktion und der Wavelet.

Die Ähnlichkeit der Wavelet mit der Funktion wird für jedes m und b berechnet und in dem 3-dimensionalen Ausgabe-Graphen in Abhängigkeit zur Zeit angegeben. Dies hat den Vorteil das man eine Darstellung des Signals in Abhängigkeit von Zeit und Frequenz erstellen kann.

Dies kann beispielsweise sinnvoll in der Überprüfung von Ampelleuchten sein. Während die Fourier-Transformation die unterschiedlichen Frequenzen der Farben grün, gelb und rot erkennt und wiedergibt ob die drei Lichter leuchten, kann die Wavelet-Transformation angeben, ob die Lampen jeweils zum richtigen Zeitpunkt leuchten [1].

- verwendet spezialisierte Funktionen: Wavelets - aus französisch Wellchen/ kleine Welle - Familie an Funktionen - jede für spezielle Anwendung - Integral unter und über X-Achse gleichgroß - Admissibility condition - endliche Fläche - finite energy - lokalisiert in Zeit

4 Code

In diesem Abschnitt wird der Python-Code zur Extraktion von harmonischen und perkussiven Komponenten aus Audiodateien unter Verwendung der Bibliothek `librosa` detailliert erklärt.

4.1 Beschreibung des Codes

Der Code beginnt mit dem Importieren der benötigten Bibliotheken, um Audio zu laden, zu verarbeiten und in neue Dateien zu schreiben:

Listing 4.1: Bibliotheken importieren

```
import os
import librosa
import soundfile as sf
import numpy as np
```

Hierbei ist `librosa` die zentrale Bibliothek zur Verarbeitung von Audiodaten, während `soundfile` für das Schreiben der resultierenden Audiodateien verwendet wird.

4.2 Hauptfunktion: `harmonic_extraction`

Die Hauptfunktion des Codes ist `harmonic_extraction`, die einen Dateipfad für die Audiodatei und einen Namen für die Ausgabedatei als Parameter erhält.

Listing 4.2: Die Funktion `harmonic_extraction`

```
def harmonic_extraction(audio_path, output_filename):
    y, sr = librosa.load(audio_path)

    y_harmonic, y_percussive = librosa.effects.hpss(y)
```

Die Funktion beginnt mit dem Laden der Audiodatei. Die Methode `librosa.load` lädt die Datei und gibt das Audiosignal `y` und die Sampling-Rate `sr` zurück. Der nächste

Schritt ist die Verwendung der Harmonic-Percussive Source Separation (HPSS) Funktion `librosa.effects.hpss(y)`, die das Audiosignal in harmonische und perkussive Komponenten zerlegt. Dies geschieht mithilfe der Fourier-Transformation.

4.3 Speichern der Ergebnisse

Nach der Zerlegung speichert der Code die beiden Komponenten in separaten Dateien:

Listing 4.3: Speichern der Komponenten

```
    current_dir = os.getcwd()

    input_audio_path = os.path.join(current_dir, 'audios', audio_path)
    output_directory = os.path.join(current_dir, 'extractedfiles')
    output_path_harmonic = os.path.join(output_directory, output_filename)
    output_path_percussive = os.path.join(output_directory, 'extracted_percussive.wav'

    os.makedirs(output_directory, exist_ok=True)

    sf.write(output_path_harmonic, y_harmonic, sr)
    print("Harmonic component saved to:", output_path_harmonic)

    y_percussive_only = y - y_harmonic

    sf.write(output_path_percussive, y_percussive_only, sr)
    print("Percussive component saved to:", output_path_percussive)
```

Das obige Codefragment erstellt das Verzeichnis `extractedfiles` und speichert darin die harmonischen und perkussiven Komponenten als separate Dateien. Die Methode `sf.write` schreibt das Audiosignal in eine Datei, die anschließend abgespielt oder analysiert werden kann.

4.4 Fourier-Transformation und `librosa`

Im Hintergrund verwendet `librosa` die Fourier-Transformation zur Analyse und Trennung der Frequenzkomponenten. Die Fourier-Transformation wandelt ein Zeitsignal in seine Frequenzkomponenten um, was es `librosa` ermöglicht, harmonische und perkussive Elemente zu isolieren.

Die Harmonic-Percussive Source Separation (HPSS) Methode analysiert das Spektrum des Audiosignals und trennt es basierend auf der Stabilität der Frequenzkomponenten

über die Zeit: Harmonische Komponenten bleiben relativ konstant, während perkussive Komponenten abrupte Änderungen im Spektrum aufweisen.

4.5 Hintergrund: Fourier-Transformation und HPSS in `librosa`

Um die Funktionsweise von `librosa` bei der Verarbeitung und Trennung von Audiosignalen nachzuvollziehen, betrachten wir die verwendeten Algorithmen, insbesondere die Short-Time Fourier Transform (STFT) und die Harmonic-Percussive Source Separation (HPSS). Diese Verfahren lassen sich mithilfe von grundlegender Signalverarbeitung in Python umsetzen.

4.5.1 Short-Time Fourier Transform (STFT)

Die STFT teilt das Audiosignal in kurze, sich überlappende Abschnitte, um das zeitliche Verhalten der Frequenzanteile zu erfassen. Dies ergibt ein Spektrogramm, das Frequenzänderungen im Zeitverlauf darstellt. Der Code für die STFT ist wie folgt:

Listing 4.4: STFT-Implementierung

```
import numpy as np

def stft(y, n_fft=2048, hop_length=512, window="hann"):
    if window == "hann":
        win = np.hanning(n_fft)
    num_frames = 1 + (len(y) - n_fft) // hop_length
    stft_matrix = np.empty((n_fft // 2 + 1, num_frames), dtype=np.complex64)

    for i in range(num_frames):
        start = i * hop_length
        frame = y[start : start + n_fft] * win
        stft_matrix[:, i] = np.fft.rfft(frame)

    return stft_matrix
```

Dieser Code berechnet das Spektrogramm des Audiosignals, indem für jedes Segment die Fourier-Transformation mit `np.fft.rfft` durchgeführt wird. Das Hann-Fenster glättet die Segmente und reduziert abrupte Übergänge zwischen den Abschnitten.

4.5.2 Harmonic-Percussive Source Separation (HPSS)

Die HPSS-Technik trennt das Spektrum des Audiosignals in harmonische und perkussive Komponenten, basierend auf der Annahme, dass harmonische Frequenzen über die Zeit stabil bleiben, während perkussive Frequenzen abrupte Änderungen aufweisen. Hierfür werden Medianfilter verwendet:

Listing 4.5: HPSS-Implementierung

```
from scipy.ndimage import median_filter

def harmonic_percussive_separation(stft_matrix, harmonic_filter_size=31, percussive_fi
    harmonic_component = median_filter(np.abs(stft_matrix), size=(1, harmonic_filter_s
    percussive_component = median_filter(np.abs(stft_matrix), size=(percussive_filter_s

    harmonic_mask = harmonic_component > percussive_component
    percussive_mask = percussive_component >= harmonic_component

    harmonic_stft = stft_matrix * harmonic_mask
    percussive_stft = stft_matrix * percussive_mask

    return harmonic_stft, percussive_stft
```

Dieser Code verwendet `median_filter`, um das Spektrum über die Zeitachse zu glätten und so harmonische und perkussive Komponenten zu trennen. Die beiden Masken trennen das Spektrum in harmonische und perkussive Anteile und erlauben eine gezielte Extraktion der einzelnen Bestandteile.

4.5.3 Inverse STFT (iSTFT)

Um das transformierte Spektrum wieder in ein Zeitsignal zu konvertieren, verwenden wir die inverse STFT:

Listing 4.6: iSTFT-Implementierung

```
def istft(stft_matrix, hop_length=512, n_fft=2048, window="hann"):
    if window == "hann":
        win = np.hanning(n_fft)
    y = np.zeros(hop_length * (stft_matrix.shape[1] - 1) + n_fft)

    for i in range(stft_matrix.shape[1]):
        frame = np.fft.irfft(stft_matrix[:, i]) * win
        start = i * hop_length
        y[start : start + n_fft] += frame

    return y
```

Die Funktion `istft` führt das Frequenzspektrum zurück in die Zeitdomäne und rekonstruiert das Audiosignal durch eine inverse Fourier-Transformation mit `np.fft.irfft`. Die Überlappungsaddierung gewährleistet eine nahtlose Rücktransformation in die Zeitdomäne.

Zusammenfassend lässt sich sagen, dass `librosa` mithilfe der STFT und HPSS das Audiosignal in harmonische und perkussive Anteile zerlegt und so die Frequenzkomponenten eines Signals analysieren und trennen kann.

Literatur

[1] Artem Kirsanov. *Wavelets: a mathematical microscope*. Abgerufen am 19.08.2024. Aug. 2022. URL: https://youtu.be/jnxqHcObNK4?si=1VZPZquUYXSuI4vN.

[2] Julian Ashbourn. *Audio Technology, Music, and Media*. 1. Berkhamsted: Springer Cham, 2020.

[3] Audioengine. *What Is Audio Compression (And Why Should You Care)?* Abgerufen am 31.08.2024. URL: https://audioengine.com/explore/what-is-audio-compression-and-why-should-you-care/.

[4] e-teaching.org. *WAV*. Abgerufen am 31.08.2024. URL: https://www.e-teaching.org/materialien/glossar/wav.

[5] Electronic Music Interactive. *Digital Representation*. Abgerufen am 31.08.2024. URL: https://pages.uoregon.edu/emi/8.php#:~:text=A%20digital%20representation%20of%20sound%20is%20a%20series%20of%20discrete,spaced%20measurements%20is%20a%20sample%20..

[6] Matthias Karmasin und Rainer Ribing. *Die Gestaltung wissenschaftlicher Arbeiten. Ein Leitfaden für Facharbeit/VWA, Seminararbeiten, Bachelor-, Master-, Magister- und Diplomarbeiten sowie Dissertationen*. 10., überarbeitete und aktualisierte Auflage. Wien: facultas, 2019.

[7] Ken. *Manipulate Audio File in Python With 6 Powerful Tips*. Abgerufen am 31.08.2024. Okt. 2021. URL: https://www.codeforests.com/2021/10/02/manipulate-audio-file-in-python/.

[8] Lehrerinnenfortbildung Baden-Württemberg. *Audioformate im Überblick*. Abgerufen am 31.08.2024. URL: https://lehrerfortbildung-bw.de/st_digital/medienwerkstatt/multimedia/audio/formate/.

[9] *Librosa*. Abgerufen am 31.08.2024. URL: https://github.com/librosa.

[10] Meinard Müller and Vlora Arifi-Müller. *Sheet Music Representations*. Abgerufen am 19.08.2024. URL: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1_SheetMusic.html.

[11] Meinard Müller. *Fundamentals of Music Processing*. 2. Erlangen: Springer Cham, 2021.

[12] NTi Audio. *Fast Fourier Transformation FFT - Grundlagen*. Abgerufen am 31.08.2024. URL: https://www.nti-audio.com/de/service/wissen/fast-fourier-transformation-fft.

[13] Roxio. *What is a WAV file?* Abgerufen am 31.08.2024. URL: https://www.roxio.com/en/file-formats/wav-file/.

[14] Stephen William. *Music Extraction*. Abgerufen am 31.08.2024. Okt. 2021. URL: https://medium.com/@swilliam.productions/music-extraction-7eb352d92bff.

Anhang

- Eventuell Git-Repo verlinken