
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	III
1 Einleitung	1
1.1 Relevanz	1
1.2 Vorgehen	2
2 Stand der Wissenschaft	3
3 Hauptteil	4
3.1 Formen der Musikdarstellungen	5
3.2 Was ist ein Ton?	5
3.3 Aufbau einer Audiodatei/ Music Representation	6
3.3.1 Durchführung der Komprimierung	6
3.3.2 Waveform Audio File Format	7
3.3.3 Andere Formen von Audio-Dateien	7
3.4 Trennung einer Tonspur in verschiedene Instrumente	7
3.4.1 Fourier Transformation	8
3.4.2 Wavelet-Transformation	10
4 Code	12
4.1 Beschreibung des Codes	12
4.2 Hauptfunktion: <code>harmonic_extraction</code>	12
4.3 Speichern der Ergebnisse	13
4.4 Fourier-Transformation und <code>librosa</code>	13
4.5 Hintergrund: Fourier-Transformation und HPSS in <code>librosa</code>	14
4.5.1 Short-Time Fourier Transform (STFT)	14
4.5.2 Harmonic-Percussive Source Separation (HPSS)	15
4.5.3 Inverse STFT (iSTFT)	15
Literatur	17
Anhang	19

Abbildungsverzeichnis

3.1	Note C4 in unterschiedlichen Darstellungen	10
-----	--	----

1 Einleitung

Musik ist zu einem Teil des täglichen Lebens 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 in die unterschiedlichen Funktionen der Frequenzen. Dies wird sowohl auf audiovisuelle als auch auf visuelle Funktionen angewendet. 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 akkustische und perkussive Instrumente getrennt. Akkustische Instrumente sind Instrumente die einen Ton erzeugen indem Menschen Kraft auf sie ausüben. Allerdings zählen Instrumente die geschlagen, geschüttelt oder geschabt werden zu den perkussiven Instrumenten.

(Frequenzen und Wellenfunktion 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 verschiedene (Gruppen von?) -> (akkustik und percussive (deutsch)) Musikinstrumente zu zerlegen. Dies kann mit weiterer Modifikation verwendet werden, um bestimmte Instrumente zum üben, zu trennen oder Hintergrundgeräusche auszublenden.

Außerdem werden in dem Projekt die Überprüfung der Funktionalität einer Ampel oder das Stimmen von Musikinstrumenten in Echtzeit behandelt. Dies sind nur Beispiele dafür, welche Anwendungsfälle es für die Fourier Transformation gibt.

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 Lesern:innen ein erstes Verständnis von WAV-Dateien vermittelt auf dem im Anschluss aufgebaut werden kann.

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

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 (?).

1. Was ist ein Ton
2. Formen der Musikdarstellung
3. Darstellung in der Mathematik
4. Vorstellung Fourier Transform
5. Ton und Fourier als Code
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)

(Bis hier muss überarbeitet werden?)

3.1 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 Notendarstellung

Eine weitere Form der Darstellung, ist die Audiodarstellung. In diesen Darstellungen werden alle Informationen der Töne als Audiosignale digital gespeichert und geteilt. 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 beibehalten.

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

- Musik Darstellung in Audiodateien - Was ist eine Note audio-/ visuell

3.2 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.

(Bild?)

- 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"

3.3 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.

- Noten sind ebenfalls musikalische Darstellung - 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 Quantisierungsgröße reduziert.

Sampling Rate

Die Sampling Rate (/Sampling-Rate?) definiert (/beschreibt?) 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 einer deutlichen Reduzierung des Speicherbedarfs führen.

(Bild einfügen?)

Quantisierungsgröße (deutscher Fachbegriff?)

Die Quantisierungsgröße beschreibt die Komprimierung pro Sample. Die möglichen Werte eines Samples sind fließend (Fachbegriff für Einheiten mit unendlich Nachkommastellen?) und werden mit der Quantisierungsgröße eingegrenzt. Damit wird die Anzahl der möglichen Werte eingegrenzt und dessen Unterschied festgelegt. Beispielsweise wird bei CDs ein 16-bit coding scheme (Deutsch?) gewählt, welches 65.536 mögliche Werte festlegt.

(Bild einfügen?)

3.3.2 Waveform Audio File Format

Waveform Audio File Format (kurz: WAV) sind Dateien, die Audioaufnahmen unkomprimiert speichern [13]. Es wird „vom englischen Wort ‚wave‘ abgeleitet, und steht in diesem Zusammenhang für Schallwelle“ [4]. Für die in diesem Projekt behandelte (Fast?) Fourier Transformation sind die Höhe der „Abtastrate des Messsystems“ und die Quantitätsgröße (siehe Kapitel...?) bedeutsam [12]. Durch die unkomprimierte Form (ist die Abtastrate (/Quantitätsgröße?) hoch und) kann die Trennung von Tonspuren (Referenz?) durchgeführt werden.

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 aufgenommene Signal in Abhängigkeit zur Zeit. Jedes der eingehenden Signale verfügt über Schwingungen in Form von einer Amplitude. Bei einer Aufnahme vermischen sich die verschiedenen Signale zu einer Amplitude und sind schwierig zu differenzieren. Beispielsweise von Hintergrundgeräuschen oder bei der Aufnahme mehrerer Musikinstrumente. Die Trennung von Instrumenten einer Tonspur wird in Forschung und Bildung behandelt (Quelle?). Die verbreitetste (Quelle?) Vorgehensweise ist die Verwendung der Fourier Transformation (Gibt es andere Vorgehensweisen?).

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 wird von der Fourier Transformation transformiert.

Entwicklung der Fourier Transformation

Die Fourier-Transformation ist eine Generalisierung der Fourierreihen. Diese können (stetige?) Funktionen in mehrere Funktionenreihen aufteilen. Bereits im 18. Jahrhundert wurden Fourierreihen für einzelne Funktionen entdeckt. Im Jahr 1822 behauptete Joseph Fourier, dass Funktionenreihen für jede Funktion existieren. Erst im 20. Jahrhundert wurden Funktionenreihen für die übrigen stetigen oder stückweise stetigen Funktionen entdeckt. Durch die Vollständigkeitsbedingungen der Funktionenreihe kann die Transformation auf jegliche Funktionen (welche Funktionen?) angewendet werden und wurde nach Fourier benannt.

Durchführung der Transformation

Die Fourier Transformation ist ein mathematisches (?) Vorgehen, indem die zeitliche Darstellung in eine frequentielle Darstellung transformiert wird. Die Transformation ermöglicht es jegliche Funktionen in eine Summe aus Sinus und Kosinus Wellen unterschiedlicher Frequenzen zu zerlegen.

- 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. 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 Samples und Quantisierungsgrößen behandelt, die für die Durchführbarkeit der Transformation relevant sind. Zudem können Komponenten bei der Durchführung (Synonym?) parametrisiert werden.

Blocklänge (/engl.?)

Ein Parameter für die Durchführung der Fourier-Transformation ist die Blocklänge (kurz: BL). Diese behandelt die Anzahl an Samples die bei der Fourier-Transformation betrachtet werden.

Umso länger die BL, desto präziser wird die Transformation durchgeführt. Jedoch bedeutet dies auch mehr Rechenaufwand und eine längere Dauer der Transformation. Bei einigen Anwendungsszenarien führt dies zu einer Abwägung zwischen Dauer und Genauigkeit der Durchführung. Beispielsweise bei einer App zum stimmen von Instrumenten und 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?).

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.

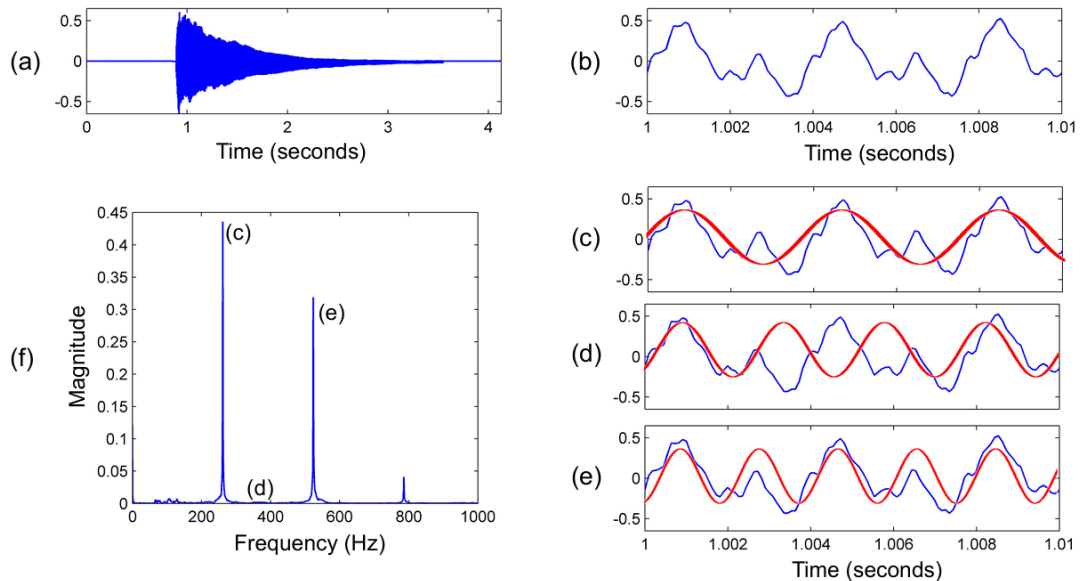


Abbildung 3.1: Note C4 in unterschiedlichen Darstellungen

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

- anschließend trennen?

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 Komponente kombiniert.

- Darstellung (nie Zeit und Frequenz)

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

3.4.2 Wavelet-Transformation

Die Wavelet-Transformation ist ein Vorgehen, das eine Funktion der zeitlichen Darstellung in eine 3(-?)dimensionale Darstellung in Abhängigkeit von Zeit und Frequenz

erstellt. Dafür werden Wellenfunktionen mit der ursprünglichen Funktion auf Übereinstimmungen verglichen. Es Wellenfunktionen gibt unterschiedliche Wellenfunktionen und sie werden Wevelets genannt. Der Name ist aus dem französischen 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). Jede Wavelet wird um die Parameter m und b ergänzt.

m: Bestimmt die Frequenz der Wavelet

b: Bestimmt den Zeitpunkt der Wavelet

Zudem hat die 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_filter_size=31):
    harmonic_component = median_filter(np.abs(stft_matrix), size=(1, harmonic_filter_size))
    percussive_component = median_filter(np.abs(stft_matrix), size=(percussive_filter_size, 1))

    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 `istfft` 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