

---

# Umsetzbarkeit der Trennung perkussiver und harmonischer Frequenzen in einer 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  
lukas.fey@smail.th-koeln.de, nicolas-friedmann@gmx.de

eingereicht bei: Prof. Dr. Lutz Köhler

Gummersbach, 10.01.2025

## **Kurzfassung/Abstract**

Diese Arbeit untersucht die Machbarkeit der Trennung perkussiver und melodischer Frequenzen in einer Wav-Datei. Im Rahmen des Projekts wird die Fourier-Transformation als Kernmethode zur Signalverarbeitung eingesetzt, um harmonische und perkussive Bestandteile von Audiosignalen zu analysieren und zu extrahieren. Ergänzend wird auf alternative Methoden wie die Wavelet-Transformation eingegangen, um deren Vor- und Nachteile gegenüber der Fourier-Transformation zu bewerten.

Die praktische Umsetzung erfolgt mit Python und der Bibliothek Librosa. Die entwickelte Methode ermöglicht es, Audiosignale in getrennte Komponenten zu zerlegen, was vielfältige Anwendungen in der Musikproduktion, Audioanalyse und -restaurierung eröffnet. Es wird gezeigt, dass Wav-Dateien aufgrund ihres geringen Informationsverlusts besonders geeignet sind, während MP3-Dateien aufgrund ihrer Verfügbarkeit ein zukünftiges Ziel für Optimierungen darstellen könnten. Abschließend reflektiert die Arbeit die Limitationen des Ansatzes und gibt einen Ausblick auf mögliche Erweiterungen, einschließlich der Nutzung maschinellen Lernens für komplexere Signaltrennungen.

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Relevanz . . . . .	1
1.2 Forschungsfrage und Hypothesen . . . . .	2
1.3 Aufbau der Arbeit . . . . .	3
1.4 Vorgehen . . . . .	3
<b>2 Stand der Wissenschaft</b>	<b>5</b>
<b>3 Theorie zur Signalverarbeitung</b>	<b>6</b>
3.1 Was ist ein Ton? . . . . .	6
3.2 Musikinstrumentgruppen . . . . .	7
3.2.1 Behandelte Instrumentgruppen . . . . .	8
3.2.2 Vorteile der Instrumentgruppen für die Signalverarbeitung . . . . .	8
3.3 Formen der Musikdarstellungen . . . . .	8
3.3.1 Musiknoten und symbolische Darstellung . . . . .	8
3.3.2 Audiodarstellung . . . . .	9
3.3.3 Einordnung in den Projektkontext . . . . .	9
3.4 Aufbau einer Audiodatei . . . . .	9
3.4.1 Durchführung der Komprimierung . . . . .	9
3.4.2 Waveform Audio File Format . . . . .	10
3.4.3 Andere Formen von Audio-Dateien . . . . .	11
3.5 Trennung einer Tonspur in verschiedene Instrumente . . . . .	11
3.5.1 Fourier Transform . . . . .	12
3.5.2 Wavelet Transform . . . . .	15
3.5.3 Wahl der Fourier Transform . . . . .	17
3.5.4 Short-Time Fourier Transform . . . . .	17
<b>4 Code</b>	<b>18</b>
4.1 Beschreibung des Codes . . . . .	18
4.2 Hauptfunktion: <code>harmonic_extraction</code> . . . . .	18
4.3 Speichern der Ergebnisse . . . . .	19
4.4 Fourier-Transformation und <code>librosa</code> . . . . .	19

4.5 Hintergrund: Fourier-Transformation und HPSS in <i>librosa</i> . . . . .	20
4.5.1 Short-Time Fourier Transform (STFT) . . . . .	20
4.5.2 Harmonic-Percussive Source Separation (HPSS) . . . . .	21
4.5.3 Inverse STFT (iSTFT) . . . . .	21
<b>5 Wav-Manipulation und Analyse</b>	<b>23</b>
5.1 Spektralanalyse der Audiodatei . . . . .	23
5.1.1 Spektrogramm der Originaldatei . . . . .	23
5.1.2 Spektrogramm der harmonischen Komponente . . . . .	24
5.1.3 Spektrogramm der perkussiven Komponente . . . . .	25
5.2 Interpretation der Ergebnisse . . . . .	25
<b>6 Fazit</b>	<b>26</b>
6.1 Beantwortung der Forschungsfrage . . . . .	26
6.2 Kritische Reflexion der eigenen Arbeit . . . . .	27
6.3 Ausblick für die Zukunft . . . . .	27
<b>Anhang</b>	<b>29</b>

## Abbildungsverzeichnis

3.1	Wellenform eines Tons . . . . .	6
3.2	Wellenform eines Tons . . . . .	7
3.3	Unterteilung in Samplingrate (oben) und Quantisierungsschrittweite <i>Quelle: fundamentals_of_music_processing</i> , S. 61 . . . . .	10
3.4	Transformation von zeitlicher zu frequenzieller Darstellung <i>Quelle: music_extraction</i> . . . . .	13
3.5	Note C4 in unterschiedlichen Darstellungen <i>Quelle: fundamentals_of_music_processing</i> , S. 41 . . . . .	14
3.6	Wavelet Beispiele in zweidimensionaler Darstellung <i>Quelle: wavelet_examples</i> . . . . .	15
3.7	Frequentielle und zeitliche Darstellung nach Wavelet Transform <i>Quelle: wavelet_transform</i> . . . . .	16
5.1	Spektrogramm der Original-Audiodatei vor der Trennung . . . . .	23
5.2	Spektrogramm der harmonischen Komponente . . . . .	24
5.3	Spektrogramm der perkussiven Komponente . . . . .	25

# 1 Einleitung

Musik ist zu einem Teil des täglichen Lebens vieler Menschen geworden (**musiknutzung**). Durch den Konsum und den wirtschaftlichen Ertrag wird an der Produktion und Analyse von Musik geforscht (**absatz**). Unter anderem gründete die Universität zu Lübeck ein eigenes Institut für Signalverarbeitung (**institute\_for\_signal\_processing**).

Ein Teil der Forschung bezieht sich auf die Trennung einer Wellenform in die einzelnen Funktionen der jeweiligen Frequenzen. Zuvor beinhaltet die Wellenform eine oder mehrere Sinus- und Kosinusfunktionen, die eine neue Funktion bilden. Dies gilt sowohl für audiovisuelle als auch für visuelle Funktionen. Beispielsweise kann die Funktion mehrere Instrumente beinhalten, die anhand der Funktion nicht identifizierbar sind. Einer der Algorithmen zur Trennung von Signalen heißt Fourier Transform 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 akustischen Instrumenten getrennt. Zwischen den jeweiligen Instrumentengruppen wird in Abschnitt 3.2 unterschieden.

## 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 gleichzeitige Töne einer Tonspur Noten zugeordnet werden, um Stimmen von Musik zu unterscheiden oder wenn einzelne Instrumente im Nachhinein bearbeitet und einzeln angehört werden (**importance\_fourier**).

Zudem ist die Audioverarbeitung lediglich ein Unterthema der Signalverarbeitung. Die Forschung an einem spezifischen Anwendungsfällen kann ebenfalls Fortschritt in einem weiteren Themenbereich der Signalverarbeitung bringen. Beispielsweise wird die Fourier Transform ebenfalls bei Laser-Doppler-Vibrometern verwendet (**Laser-Doppler-Vibrometer**).

In Zeiten von zunehmend digital produzierter Musik und Verbreitung von Informatik stellt sich die Frage, wie Musiksignale digital aufgebaut sind und wie mit ihnen gearbeitet wird. Im Kontext dieser Arbeit wird die Fourier Transform verwendet, um eine Tondatei in Gruppen von harmonischen und perkussiven Musikanstrumenten zu zerlegen. Dies kann mit weiterer Modifikation verwendet werden, um bestimmte Instrumente für das Üben zu trennen oder Hintergrundgeräusche auszublenden.

## **1.2 Forschungsfrage und Hypothesen**

Dieses Projekt dient dem Einstieg in die Thematik der Signalverarbeitung. Dabei werden Audiosignale und deren Entstehung behandelt. Es wird überprüft, ob die Trennung von Percussive und Melodic (klein? auf Deutsch? Noch nicht einheitlich) Frequenzen in einer Wav-Datei umsetzbar ist.

**Bei der Bearbeitung ergeben sich weitere Fragen zu einzelnen Unterthemen:**

- Wie entstehen Frequenzen und wie kann man zwischen ihnen unterscheiden?
- Was zeichnet die behandelten Instrumentgruppen aus und warum wurden sie gewählt?
- Welche Audio-Speicherformen bestehen und sind für das Projekt geeignet?
- Existieren alternative Methoden zur Fourier Transform?

**Zudem wird das Projekt mit den folgenden Annahmen bearbeitet, die während des Projekts widerlegt werden können:**

- Perkussive und Melodic Instrumente können aufgrund der unterschiedlichen Frequenzen im Rahmen dieses Projekts getrennt werden können.
- Wav-Dateien sind besonders geeignet für die Trennung von Instrumenten aufgrund des geringen Informationsverlusts.
- Das implementierte Programm wird aufgrund vorhandener Bibliotheken möglichst simpel gehalten und kann von Leser:innen nachvollzogen, übernommen und erweitert werden.

## **1.3 Aufbau der Arbeit**

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

1. Was ist ein Ton
2. Formen der Audiodarstellung
3. Vorstellung der Fourier Transform
4. Alternative Methode zur Fourier Transform
5. Trennung von Instrumenten im Code

Um einen Algorithmus zu implementieren, ist es hilfreich die unterschiedlichen Musikdarstellungen kennenzulernen. Diese werden im Anschluss am Aufbau eines Tons dargestellt. Für eine effektive Fourier Transform gibt es Voraussetzungen, die nur von wenigen Musikdarstellungen erfüllt werden.

Anschließend werden unterschiedliche Methoden zur Transformation - inklusive der Fourier Transform - eines Signals behandelt, da es relevant ist ein Verständnis für die Transformationen zu entwickeln. Dies hilft nachzuvollziehen, warum die Fourier Transform geeignet ist und zukünftig den Einsatz neuer Transformations-Methoden abzuwägen.

Abschließend wird ein Algorithmus beispielhaft implementiert, der die Trennung von perkussiven und melodischen Instrumenten mittels Fourier Transform durchführt. Anhand des Codes können Merkmale der Transformation einzeln und im Kontext des gesamten Prozesses wiedergegeben werden.

## **1.4 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 Transform.

Durch die Dokumentation wird anderen Leser:innen ein erstes Verständnis von Signalverarbeitung vermittelt auf dem 4aufgebaut werden kann. Dabei werden unterschiedliche Dateiformate, Methoden zur Transformation von Signalen und die Trennung unterschiedlicher Signale behandelt (siehe: Kapitel 3).

Das Projekt dokumentiert anschließend die Implementierung eines Algorithmus zur Trennung der harmonischen und perkussiven Tongruppen einer WAV-Datei mittels Fourier Transform, sowie das schreiben auf zwei getrennte WAV-Dateien mithilfe Inverse Fourier Transform (siehe: Kapitel 4).

## 2 Stand der Wissenschaft

Die Trennung von Musikinstrumenten ist ein praktischer Anwendungsfall zur Separation von Signalen. Es entwickelte sich aus dem Gebiet der Sprachverarbeitung und übernimmt einige Techniken (**Mueller\_2011**). Dabei liegen in einer Aufnahme mehrere Signale in Form von einer Funktion (hier: Wellenform) vor. In der Wellenform sind die einzelnen Signale schwierig zu identifizieren.

Bei dem Vorgehen zur Trennung von Signalen wird eine Funktion von der Abhängigkeit zur Zeit in die Abhängigkeit zur Frequenz umgeformt. Anhand der Frequenz können unterschiedliche Signale identifiziert werden. Unter anderem werden die Frequenzen unterschiedlichen Instrumenten oder Musiknoten zugeordnet.

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

In diesem Projekt wird die Fourier Transform angewendet, die als verbreitet und effektiv gilt (**fourier\_transform\_importance**). Es wird überprüft, ob die Implementierung und Anwendung ohne Vorkenntnisse und anspruchsvolle Hardware umsetzbar ist. Eine weitere verbreitete Methode ist die Wavelet Transform (**Guo\_2022**). Diese wird in Abschnitt 3.5.2 erläutert, um die Entscheidung für die Fourier Transform zu begründen.

## 3 Theorie zur Signalverarbeitung

Vor dem Einstieg in die Signalverarbeitung wird Hintergrundwissen benötigt. Für die Durchführung der Fourier Transform werden gleichzeitig aufgenommen Signale wieder in die ursprünglichen Signale getrennt. Die Komplexität erschließt sich durch die Darstellung vermischter Signale und dessen Schwierigkeit einzelne Signale wiederzufinden. Dafür werden zuvor der Aufbau einzelner Signale behandelt, um zu erkennen wie Frequenzen dargestellt und abgelesen werden.

### 3.1 Was ist ein Ton?

Töne entstehen durch die Vibratoren eines Gegenstandes, die Schallwellen erzeugen. Das Gehör kann diese Schwingungen der Luft wahrnehmen (**signaltoene**). Der Luftdruck einer Schallwelle wird graphisch als eine Sinus- oder Cosinusfunktion dargestellt (siehe: Abbildung 3.1).

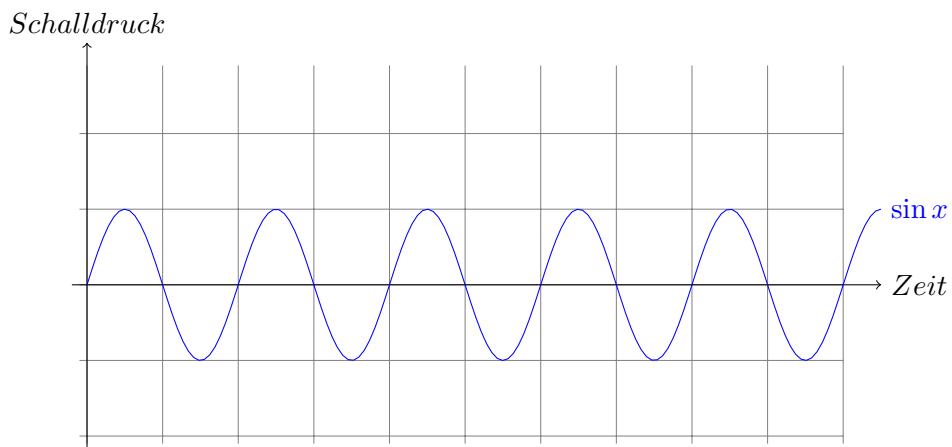


Abbildung 3.1: Wellenform eines Tons

Anhand der Wellenform kann die Frequenz eines Tons abgeleitet und in Hertz (kurz: hz) angegeben werden. Die Frequenz gibt die Anzahl an Zyklen des Tons pro Sekunde

an. Wenn beispielsweise Abbildung 3.1 eine Sekunde darstellt, entspricht die Frequenz 5hz, da sie fünf symmetrische Wellen aufweist.

Diese Vibration kann durch unterschiedliche Gegenstände erzeugt werden. Dazu zählen Becken eines Schlagzeugs, Saiten eines Kontrabasses oder die Stimmbänder einer Person. In diesem Projekt werden die Perkussionsinstrumente von den restlichen Instrumenten einer Wav-Datei getrennt.

Bei der Darstellung mehrerer Signale werden einzelne Signale zu einer gemeinsamen Wellenform addiert. In Abbildung 3.2 werden eine 5hz (rot) und eine 2,5hz (blau) Wellenform gleichzeitig aufgenommen. Der Summe der beiden Signale (grün) sind die einzelnen Bestandteile kaum abzulesen. Die Herausforderung Frequenzen der Bestandteile zu identifizieren wird mittels Transformationen bewältigt.

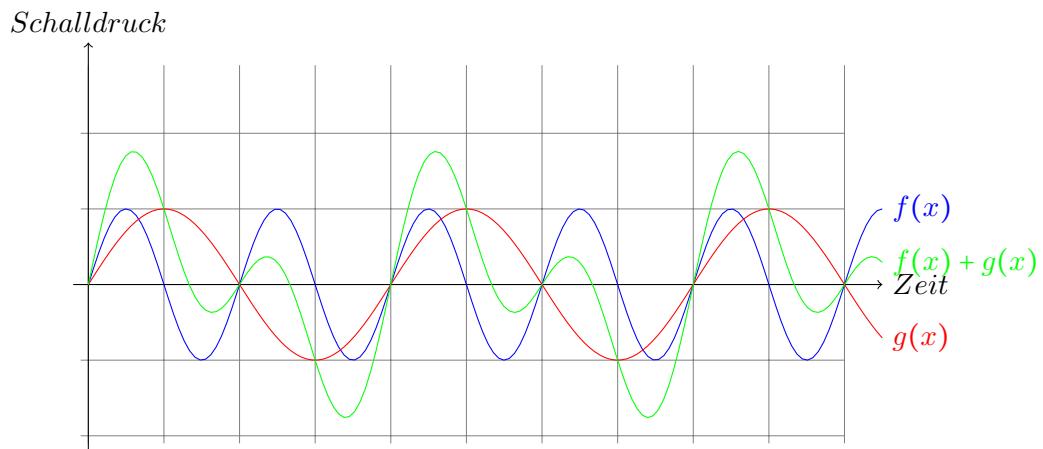


Abbildung 3.2: Wellenform eines Tons

## 3.2 Musikinstrumentgruppen

Musikinstrumente können in mehrere Gruppen aufgeteilt werden. Diese unterscheiden sich voneinander in der Entstehung und dem Klang der Töne. In diesem Projekt werden akustische Instrumente behandelt, die in harmonische und perkussive Instrumente aufgeteilt werden. Darüberhinaus existieren noch elektrische und digitale Instrumente. Sie benötigen entweder Elektrizität oder einen Computerprozessor, um wie vorgesehen zu funktionieren. Allerdings werden sie in diesem Projekt nicht behandelt.

### 3.2.1 Behandelte Instrumentgruppen

**Akkustische Instrumente** erzeugen einen Ton indem Menschen Kraft auf sie ausüben. Sie benötigen keinen Strom, keine Verstärkung und keinen Computerprozessor, um einen Ton zu erzeugen und sind die ältesten Instrumente. Sie können in perkussive und harmonische Instrumente unterteilt werden.

**Perkussive Instrumente** werden geschlagen, geschüttelt oder geschabt und sind eine spezielle Form der akkustischen Instrumente. Die übrigen akkustischen Instrumente sind **harmonisch (acoustic\_electric\_digital\_instruments)**. In diesem Projekt wird versucht perkussive und harmonische Instrumente zu trennen.

### 3.2.2 Vorteile der Instrumentgruppen für die Signalverarbeitung

Akkustische Musikinstrumente zeichnet deren natürliche Entstehung aus. Bei der Entstehung von Tönen werden zusammenhängende Schallwellen erzeugt, deren Verlauf kontinuierlich ist. Dadurch werden akkustische Instrumente effektiver voneinander getrennt, da deren Verlauf nachvollziehbar ist.

Zudem sind perkussive und harmonische Musikinstrumente geeignet für die Trennung voneinander. Die unterschiedlichen Signale zeichnen sich durch die Extreme der Ausschläge der Frequenzen aus. Während perkussive Instrumente bei der Entstehung einen deutlichen Ausschlag aufweisen, verlaufen harmonische Instrumente einheitlich mit weniger extremen Ausschlägen (siehe Vergleich von Abbildung 5.3 und Abbildung 5.2).

## 3.3 Formen der Musikdarstellungen

Musik kann unterschiedlich dargestellt werden und es wird je nach Bedarf eine andere Form der Musikdarstellung benötigt. Unterteilt wird in Musiknoten, symbolische Darstellungen und Audiodarstellungen.

### 3.3.1 Musiknoten und symbolische Darstellung

Musiknoten sind eine formale Sprache, die vorgibt wie ein Musikstück gespielt wird (**sheet\_music\_representations**). Bei symbolischen Darstellungen werden eindeutige Entitäten definiert, die von einem Computer übersetzt werden. Beispielsweise wird der Musical Instrument Digital Interface (kurz: MIDI) Standard verwendet, um Informationen eines gespielten Tons möglichst detailliert zu speichern und abzurufen.

### 3.3.2 Audiodarstellung

Eine weitere Form der Darstellung, ist die Audiodarstellung. In dieser Darstellung werden die Informationen von Tönen als Audiosignale digital gespeichert und geteilt. Es werden die Schallwellen eines Tons (siehe Abschnitt 3.1) aufgenommen und digital als eine Wellenform 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 beinhalten (**fundamentals\_of\_music\_processing**, S. 1ff).

### 3.3.3 Einordnung in den Projektkontext

Bei der Trennung von Musikinstrumenten wird aufgenommene Musik in der digitalen Darstellung behandelt. Diese Darstellung erschwert es unterschiedliche Audiosignale zu trennen und deren ursprüngliche Töne zu reproduzieren, dessen verbreitetste Darstellungsform MP3 ist (**mp3\_most\_popular**). In diesem Projekt werden jedoch Wav-Dateien behandelt. Die Unterschiede, sowie Vor- und Nachteile der Audiodarstellungen werden in Abschnitt 3.4.3 behandelt.

## 3.4 Aufbau einer Audiodatei

Für die Aufnahme von Audiosignalen werden analoge Signale in eine digitale Form von Schall umgewandelt. Ein analoges Signal basiert auf kontinuierlichen und konstanten Spannungsschwankungen, die den verursachten Luftdruckschwankungen entsprechen (**digital\_representation**). In der digitalen Form werden die Spannungsschwankungen als Bitstreams gespeichert und je nach Audio-Format komprimiert.

### 3.4.1 Durchführung der Komprimierung

Die Komprimierung von Audiodateien wird meistens verwendet, um mehr Musik auf einem Datenträger speichern zu können (**what\_is\_audio\_compression**). Für die Komprimierung einer Aufnahme wird diese in eine Samplingrate (auch: Abtastrate) und eine Quantisierungsschrittweite reduziert.

#### Samplingrate

Die Samplingrate definiert die Anzahl von gespeicherten Signalen pro Sekunde. Dies reduziert die benötigten gespeicherten Daten einer durchgängigen Aufnahme auf

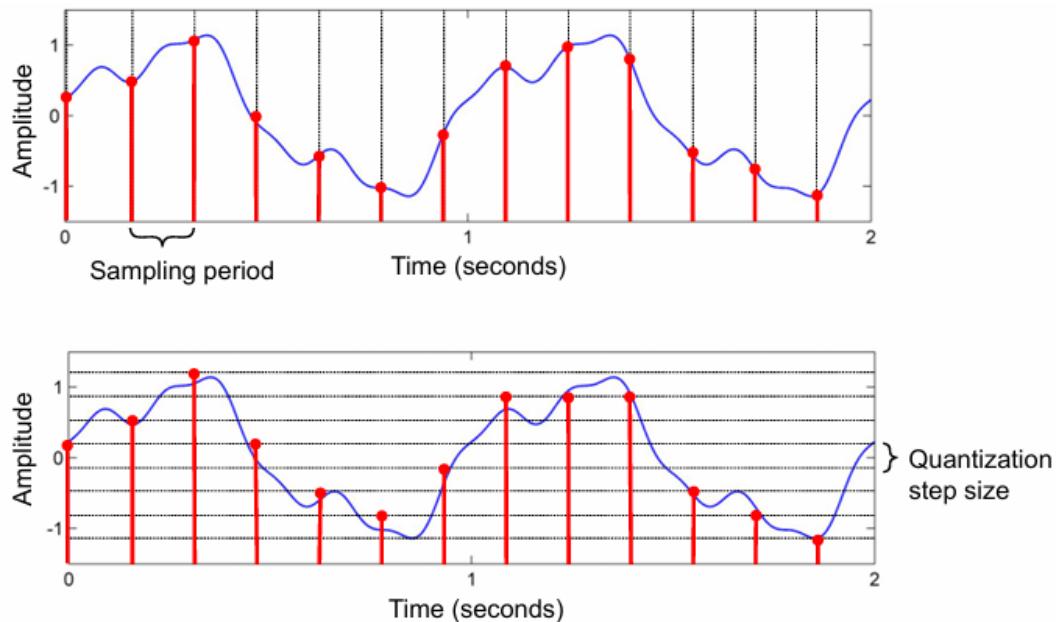


Abbildung 3.3: Unterteilung in Samplingrate (oben) und Quantisierungsschrittweite  
 Quelle: [fundamentals\\_of\\_music\\_processing](#), S. 61

mehrere Blöcke (siehe: Abbildung 3.3). Beispielsweise werden auf einer CD 44.100 Werte pro Sekunde gespeichert (kurz: 44.1 kHz), dessen Übergänge kaum wahrnehmbar sind, jedoch bereits zu einer deutlichen Reduzierung des Speicherbedarfs führen.

### 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 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.

### 3.4.2 Waveform Audio File Format

Das Waveform Audio File Format (kurz: Wav-Datei) speichert Audioaufnahmen unkomprimiert ([what\\_is\\_a\\_wav\\_file](#)). Der Begriff leitet sich „vom englischen Wort ‚wave‘“ (**wav**) für Schallwelle ab. Für die in diesem Projekt behandelte Fourier Transform sind insbesondere die „Abtastrate  $fs$  des Messsystems“ und die Quantisierungs-

schriftweite (siehe Abschnitt 3.4.1) entscheidend (**FFT\_grundlagen**). Aufgrund der unkomprimierten Speicherung (und der damit verbundenen hohen Abtastrate und Quantisierungsschrittweite) lässt sich eine klare Trennung von Tonspuren ermöglichen.

### **3.4.3 Andere Formen von Audio-Dateien**

Außer Wav-Dateien gibt es noch einige weitere verbreitete Speicherformate für Audio. Jede dieser Speicherformen hat Vor- und Nachteile und eigene Anwendungsszenarien. Allerdings stellt sich heraus, dass keine Speicherform einen geringeren Informationsverlust als Wav-Dateien aufweist.

MP3: Verwendet standardisiertes Komprimierungsverfahren und benötigt relativ wenig Speicherplatz bei vergleichsweise hoher Qualität.

WMA: Speziell für Microsoft entwickeltes Dateiformat ebenfalls mit sowohl hoher Kompression als auch guter Qualität.

AAC: Ist eine Weiterentwicklung der Entwickler:innen von MP3 mit verbessertem Verhältnis aus Komprimierung und Qualität.

OGG: Wurde als frei-verfügbare Alternative zu MP3 entwickelt.

FLAC: Steht ebenfalls patentfrei zur Verfügung und implementiert ein Verfahren zum Kodieren und Dekodieren der Daten.

RM: Steht für Real Media und beinhaltet das Real Audio Format mit Fokus auf guter Qualität, trotz Komprimierung.

(**audioformate\_im\_Aijberblick**)

## **3.5 Trennung einer Tonspur in verschiedene Instrumente**

Eine Aufnahme speichert das eingehende Signal in Abhängigkeit zur Zeit. Dabei weist jedes Signal charakteristische Schwingungen in Form einer Wellenform auf. Während einer Aufnahme überlagern sich mehrere Signale zu einer gemeinsamen Wellenform, was ihre Unterscheidung erschwert (siehe: Abbildung 3.2). Dies tritt insbesondere bei Hintergrundgeräuschen oder bei der parallelen Aufnahme mehrerer Musikinstrumente auf.

Die Trennung von Signalen ist ein Thema, das in Forschung und Bildung intensiv behandelt wird (siehe: Kapitel 2). Eine der am häufigsten verwendete Methode

ist die Fourier Transform (**fundamentals\_of\_music\_processing**, S.39), die es ermöglicht, ein Signal in seine Frequenzkomponenten zu zerlegen. Diese wird auch bei der Audioverarbeitung für die Trennung von Instrumenten innerhalb einer Tonspur verwendet.

Neben der Fourier Transform werden weitere Ansätze wie die Wavelet Transform, die Short-Time Fourier Transform (kurz: 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.

### 3.5.1 Fourier Transform

Die Fourier Transform 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 Transform transformiert.

#### Entwicklung der Fourier Transform

Die Fourier Transform 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 Vollständigkeit der Funktionenreihe lässt sich die Fourier Transform auf eine Vielzahl von Funktionen anwenden, einschließlich periodischer und nicht-periodischer Funktionen, und erhielt ihren Namen zu Ehren von Joseph Fourier.

#### Durchführung der Transformation

Die Fourier Transform ist ein mathematisches Verfahren, bei dem ein Signal aus dem Zeitbereich in den Frequenzbereich transformiert wird (siehe: Abbildung 3.4). Die Transformation ermöglicht es, beliebige periodische und stückweise stetige Funktionen als Summe von Sinus- und Kosinuswellen unterschiedlicher Frequenzen darzustellen.

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 zurückgeformt 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.

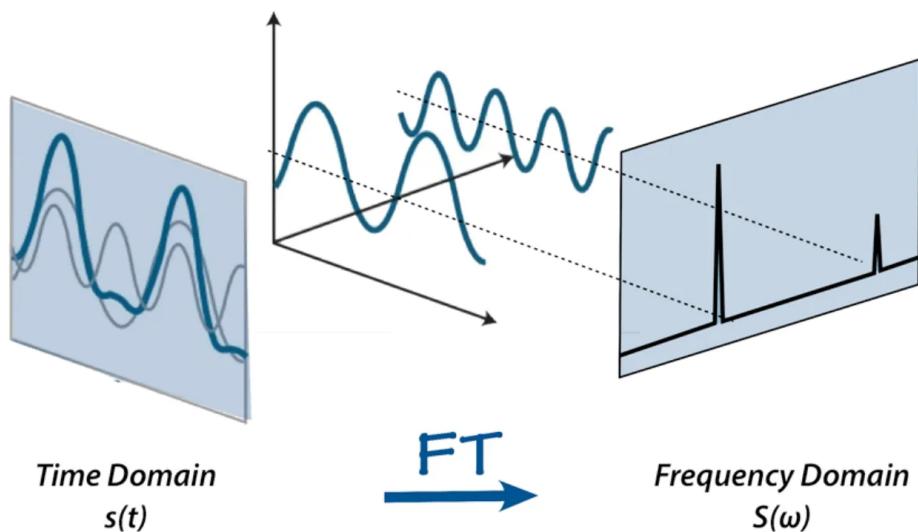


Abbildung 3.4: Transformation von zeitlicher zu frequenzieller Darstellung  
Quelle: **music\_extraction**

#### Durchführung am Beispiel einer Musiknote

In diesem Beispiel (aus **fundamentals\_of\_music\_processing**, S.40f) wird eine Note auf einem Piano gespielt und durch die Transformation in eine frequentielle Darstellung umgeformt, in der ein gespielter Ton erkannt wird.

Die Aufnahme des Tons ist in Abbildung 3.5(a) zu erkennen. Für die Transformation wird ein Ausschnitt von 10ms verwendet, um den Rechenaufwand zu reduzieren und den Vorgang beispielhaft zu verdeutlichen.

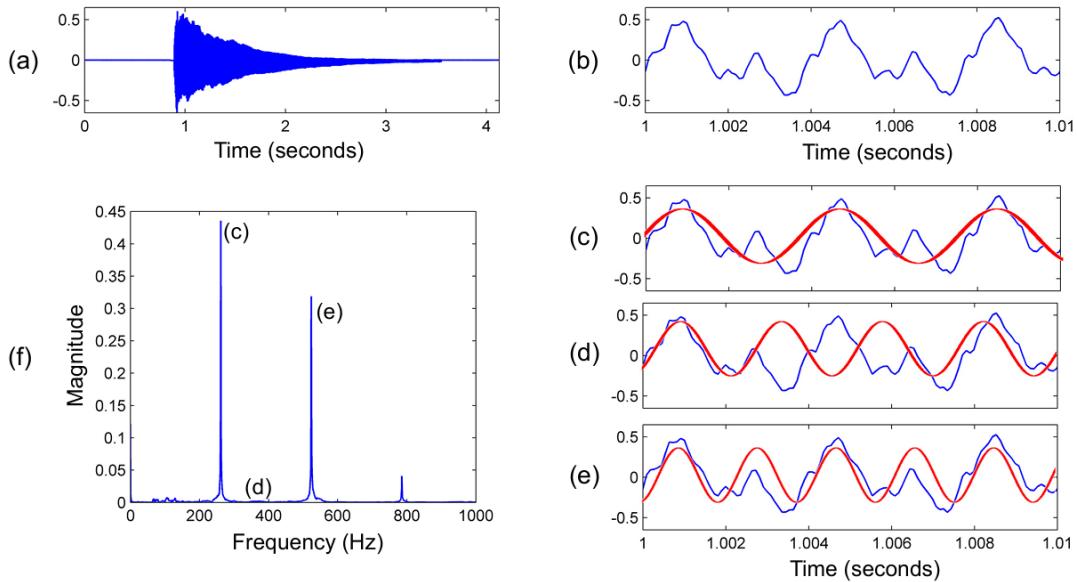


Abbildung 3.5: Note C4 in unterschiedlichen Darstellungen

Quelle: [fundamentals\\_of\\_music\\_processing](#), S. 41

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

In Abbildung 3.5(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 3.5(e)) eine hohe Übereinstimmung erkannt. Dies entspricht ungefähr der Frequenz des zweiten Teiltons der Note C4.

### Nachteile der Fourier Transform

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

Außerdem wird bei der Fourier Transform die ganze Datei bearbeitet. Dies führt bei größeren Dateien zu großem Rechenaufwand des Prozessors und zu Ungenauigkeiten in der Durchführung, da kleinere Abschnitte ignoriert werden.

### 3.5.2 Wavelet Transform

Die Wavelet Transform ist ein Verfahren, das eine zeitliche Darstellung einer Funktion in eine dreidimensionale Darstellung in Abhängigkeit von Zeit und Frequenz überführt. Dabei werden sogenannte Wavelets - spezielle Wellenfunktionen - mit der ursprünglichen Funktion verglichen, um Übereinstimmungen zu finden. Der Begriff „Wavelet“ stammt aus dem Französischen und bedeutet „kleine Welle“ oder „Wellchen“.

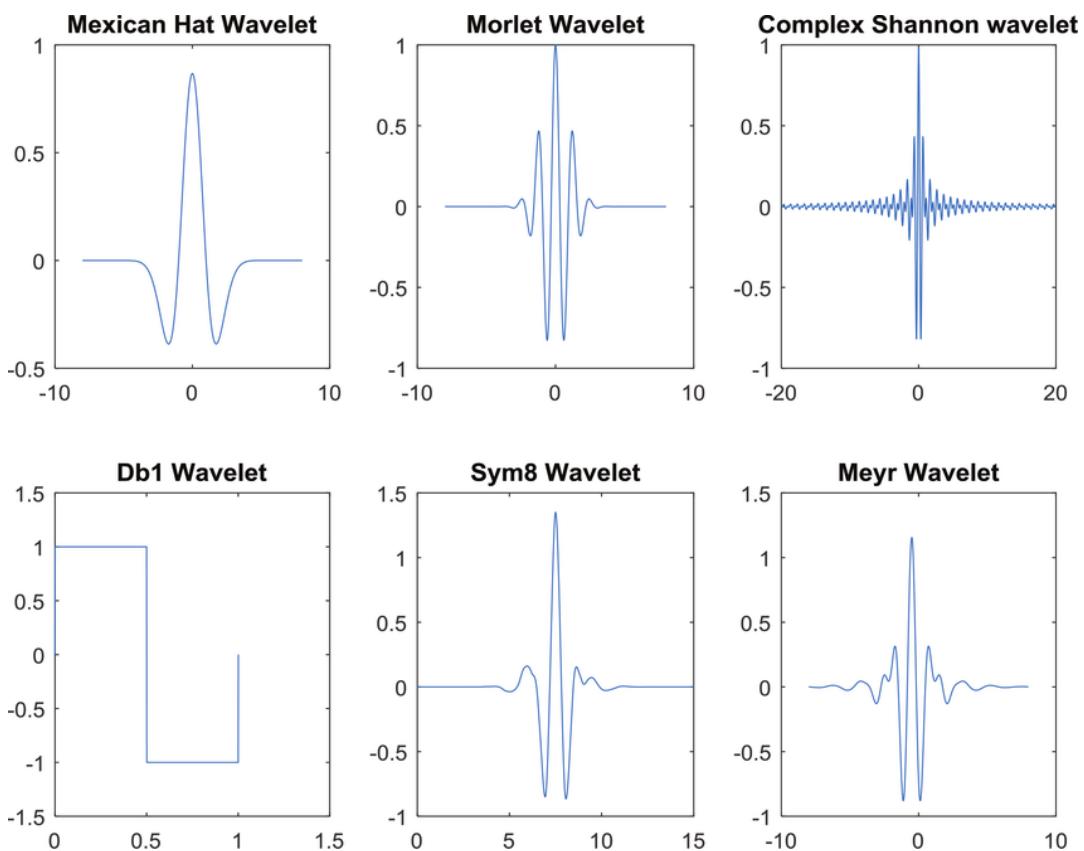


Abbildung 3.6: Wavelet Beispiele in zweidimensionaler Darstellung

Quelle: [wavelet\\_examples](#)

Im Gegensatz zur ursprünglichen Funktion haben die Wavelets eine endliche Fläche (auch: finite energy), was sie begrenzt und lokalisiert. Eine weitere Bedingung für Wavelets ist, dass ihr Integral null ergibt, d.h., dass die Fläche über und unter der X-Achse gleich groß ist (Admissibility condition). Unterschiedliche Wavelets verfügen über unterschiedliche Anwendungsszenarien.

**Jedes Wavelet wird durch die Parameter m und b ergänzt:**

m: Bestimmt die Frequenz des Wavelets

b: Bestimmt den Zeitpunkt des Wavelets

Zudem besitzt das Wavelet einen realen und einen imaginären Teil. Durch die Berücksichtigung des imaginären Teils entsteht eine dreidimensionale Darstellung des Signals. Bei der Wavelet Transform wird sowohl der reale als auch der imaginäre Teil des Wavelets mit der Funktion korreliert (Mathematik: Korrelation), um die Ähnlichkeit der Funktion und des Wavelets zu berechnen. Diese Ähnlichkeit wird für jedes m und b ermittelt und in einem dreidimensionalen Ausgabe-Graphen in Abhängigkeit von der Zeit und der Frequenz dargestellt (siehe: **wavelet\_transform**).

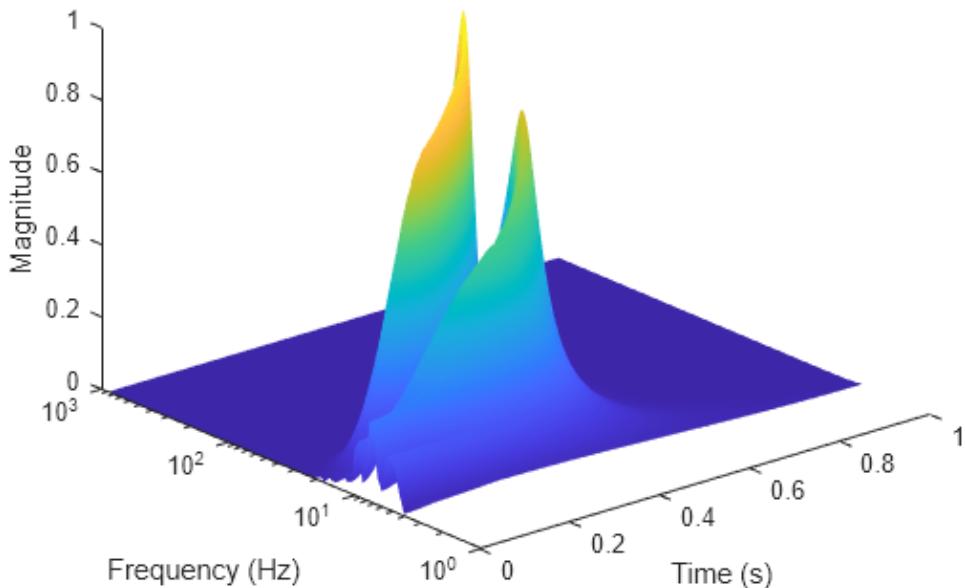


Abbildung 3.7: Frequentielle und zeitliche Darstellung nach Wavelet Transform

Quelle: [wavelet\\_transform](#)

Ein Anwendungsfall ist die Überprüfung von Ampelleuchten. Während die Fourier Transform die verschiedenen Frequenzen der Farben Grün, Gelb und Rot erkennt, um festzustellen, ob die Lampen leuchten, erlaubt die Wavelet Transform zusätzlich die Angabe, ob die Lichter zu den richtigen Zeitpunkten aufleuchten (**wavelets**).

### 3.5.3 Wahl der Fourier Transform

Für die Trennung der Instrumente einer Wav-Datei wurde die Fourier Transform gewählt, trotz einiger Alternativen. Die Fourier Transform ist eines der meistverwendeten Werkzeuge der Signalverarbeitung (**fundamentals\_of\_music\_processing**, S.39). Der größte Nachteil der Fourier Transform ist, dass nicht gleichzeitig die Zeit und die Frequenz Domäne dargestellt werden können.

Jedoch reicht dies bei der Trennung von Musikinstrumenten, da die Darstellung zum Schluss wieder in die zeitliche Domäne umgeformt wird, um das Ergebnis in eine Wav-Datei zu überführen. Außerdem verliert die Fourier Transform wenig Informationen durch die klare Trennung von zeitlicher und frequentieller Darstellung (**Parsons\_2000**).

Allerdings ist die Fourier Transform bei größeren Dateien aufwändiger und fehleranfälliger, falls die ganze Datei transformiert wird. Jedoch wird diese Limitierung durch die Verwendung der Short-Time Fourier Transform reduziert (**Prashanth\_2017**).

### 3.5.4 Short-Time Fourier Transform

Die Short-Time Fourier Transform (auch: STFT) basiert auf der Fourier Transform, dessen Nachteile zunehmender Aufwand und fehleranfälligkeit bei der Transformation größerer Dateien beinhaltet (siehe: Abschnitt 3.5.1). Damit ist es eins der wichtigsten Tools in der Audioverarbeitung (**fundamentals\_of\_music\_processing**, S.110).

Stattdessen teilt die STFT eine Datei in mehrere kleine Pakete, deren Transformation effizienter und effektiver durchgeführt werden können. Anschließend werden die Übergänge der Pakete geglättet, um Unregelmäßigkeiten zu vermeiden. In Abschnitt 4.5.1 wird die praktische Umsetzung mittels Programmcode dargestellt und die Funktionalität erläutert.

## 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:

```
1 import os
2 import librosa
3 import soundfile as sf
4 import numpy as np
```

Listing 4.1: Bibliotheken importieren

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.

```
1 def harmonic_extraction(audio_path, output_filename):
2     y, sr = librosa.load(audio_path)
3
4     y_harmonic, y_percussive = librosa.effects.hpss(y)
```

Listing 4.2: Die Funktion `harmonic_extraction`

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:

```

1 current_dir = os.getcwd()
2
3 input_audio_path = os.path.join(current_dir, 'audios', audio_path)
4 output_directory = os.path.join(current_dir, 'extractedfiles')
5 output_path_harmonic = os.path.join(output_directory,
6     ↴ output_filename)
6 output_path_percussive = os.path.join(output_directory,
7     ↴ 'extracted_percussive.wav')
8
9 os.makedirs(output_directory, exist_ok=True)
10
11 sf.write(output_path_harmonic, y_harmonic, sr)
12 print("Harmonic component saved to:", output_path_harmonic)
13
14 y_percussive_only = y - y_harmonic
15
16 sf.write(output_path_percussive, y_percussive_only, sr)
17 print("Percussive component saved to:", output_path_percussive)

```

Listing 4.3: Speichern der Komponenten

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:

```

1 import numpy as np
2
3 def stft(y, n_fft=2048, hop_length=512, window="hann"):
4     if window == "hann":
5         win = np.hanning(n_fft)
6     num_frames = 1 + (len(y) - n_fft) // hop_length
7     stft_matrix = np.empty((n_fft // 2 + 1, num_frames),
8                           dtype=np.complex64)
9
10    for i in range(num_frames):
11        start = i * hop_length
12        frame = y[start : start + n_fft] * win
13        stft_matrix[:, i] = np.fft.rfft(frame)
14
15    return stft_matrix

```

Listing 4.4: STFT-Implementierung

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:

```

1  from scipy.ndimage import median_filter
2
3  def harmonic_percussive_separation(stft_matrix,
4      ↴ harmonic_filter_size=31, percussive_filter_size=31):
5      harmonic_component = median_filter(np.abs(stft_matrix),
6          ↴ size=(1, harmonic_filter_size))
7      percussive_component = median_filter(np.abs(stft_matrix),
8          ↴ size=(percussive_filter_size, 1))
9
10     harmonic_mask = harmonic_component > percussive_component
11     percussive_mask = percussive_component >= harmonic_component
12
13     harmonic_stft = stft_matrix * harmonic_mask
14     percussive_stft = stft_matrix * percussive_mask
15
16     return harmonic_stft, percussive_stft

```

Listing 4.5: HPSS-Implementierung

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:

```

1  def istft(stft_matrix, hop_length=512, n_fft=2048, window="hann"):
2      if window == "hann":
3          win = np.hanning(n_fft)
4      y = np.zeros(hop_length * (stft_matrix.shape[1] - 1) + n_fft)
5

```

```
6     for i in range(stft_matrix.shape[1]):  
7         frame = np.fft.irfft(stft_matrix[:, i]) * win  
8         start = i * hop_length  
9         y[start : start + n_fft] += frame  
10    return y
```

Listing 4.6: iSTFT-Implementierung

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

# 5 Wav-Manipulation und Analyse

In diesem Kapitel wird anhand von Spektrogrammen, die mit Python und der Bibliothek `librosa` generiert wurden, die Analyse und Visualisierung der harmonischen und perkussiven Komponenten des Audiosignals demonstriert. Der Fokus liegt auf den Frequenzverteilungen und der Amplitudenhüllkurve.

## 5.1 Spektralanalyse der Audiodatei

Im Folgenden wird die Frequenzverteilung des Audiosignals vor und nach der Trennung in harmonische und perkussive Komponenten dargestellt. Diese Analyse ermöglicht es, die Struktur des Signals und die Unterschiede zwischen den beiden Komponenten zu visualisieren.

### 5.1.1 Spektrogramm der Originaldatei

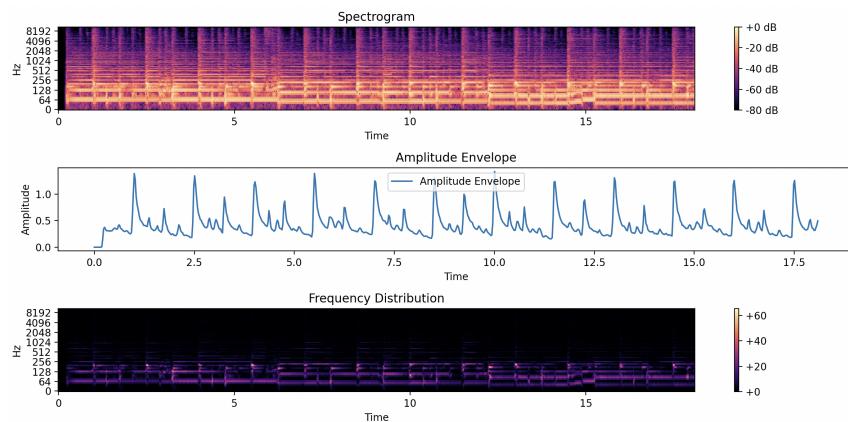


Abbildung 5.1: Spektrogramm der Original-Audiodatei vor der Trennung

Abbildung 5.1 zeigt das Spektrogramm des ungetrennten Audiosignals. Die Farbskala stellt die Amplituden der Frequenzanteile dar, wobei hellere Farben höhere Amplituden

repräsentieren. Dieses Spektrogramm stellt das gesamte Frequenzspektrum dar, das sowohl harmonische als auch perkussive Elemente enthält.

### 5.1.2 Spektrogramm der harmonischen Komponente

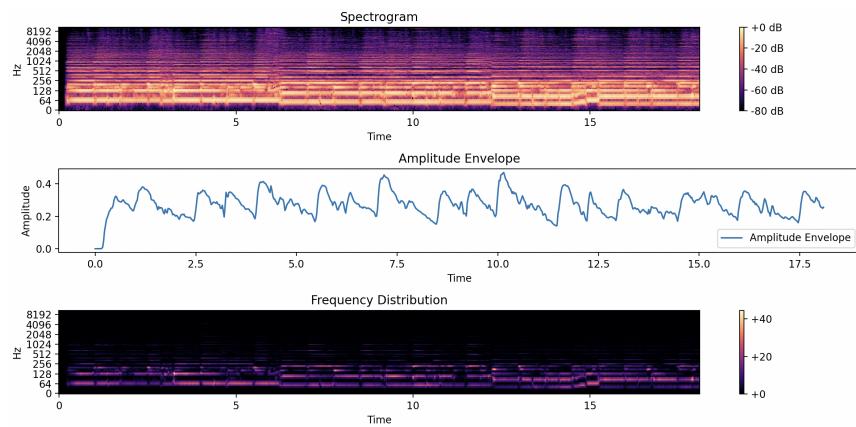


Abbildung 5.2: Spektrogramm der harmonischen Komponente

Abbildung 5.2 zeigt das Spektrogramm der harmonischen Komponente nach der Trennung. Die harmonischen Komponenten des Signals sind Frequenzen, die stabil und langanhaltend sind, was typisch für melodische oder gesangliche Elemente ist. Man sieht eine gleichmäßige Verteilung im Frequenzbereich mit weniger plötzlichen Amplitudenänderungen.

### 5.1.3 Spektrogramm der perkussiven Komponente

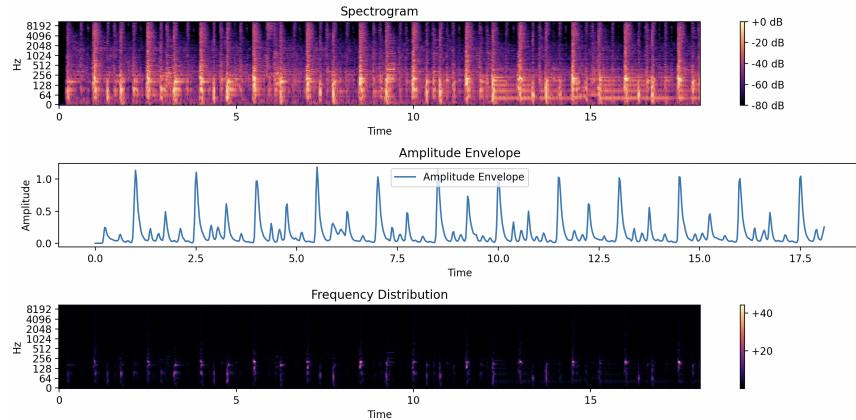


Abbildung 5.3: Spektrogramm der perkussiven Komponente

Abbildung 5.3 zeigt das Spektrogramm der perkussiven Komponente. Die perkussiven Elemente zeigen eine charakteristische Struktur, da sie Frequenzen darstellen, die plötzliche Änderungen aufweisen, typischerweise durch kurze, abrupte Schläge oder rhythmische Akzente gekennzeichnet.

## 5.2 Interpretation der Ergebnisse

Durch die Trennung der Audiodatei in harmonische und perkussive Komponenten wird die Spektralanalyse differenziert. Harmonische Spektren zeigen konstante, langanhaltende Frequenzen, während perkussive Spektren kurze, intensive Peaks aufweisen. Dies erlaubt eine gezielte Analyse und Bearbeitung der musikalischen Elemente, die für verschiedene Audioverarbeitungsaufgaben wie die Musikproduktion, Remixing oder Audio-Restaurierung wertvoll ist.

# **6 Fazit**

Um verschiedene Musikanstrumentgruppen zu trennen, wird zuvor die Theorie bearbeitet. Hierfür wird der Prozess von der Entstehung eines Tons bis zur Trennung einer Wav-Datei herausgearbeitet. Dabei werden Alternativen zu dem vorgegebenen Vorgehen verglichen und evaluiert. Es gibt alternative Speicherformen von Audio-Dateien (siehe: Abschnitt 3.4.3) und alternative Methoden zur Transformation einer Audio-Datei (Synonym?) (siehe: Abschnitt 3.5.2).

Anschließend wird die behandelte Theorie anhand einer praktischen Durchführung wiedergegeben. Die Ausarbeitung der Theorie und das erworbene Verständnis für das Themengebiet helfen bei der praktischen Umsetzung. Damit wird es Leser:innen ermöglicht einen Basiscode zu implementieren, den sie nachvollziehen und darauf aufbauen können. Außerdem können durch die Umsetzung des Codes die eingangs gestellte Forschungsfrage und Hypothesen (siehe: Abschnitt 1.2) beantwortet werden.

## **6.1 Beantwortung der Forschungsfrage**

Durch die Erarbeitung von verbundenem Hintergrundwissen wird ein Algorithmus zur Beantwortung der Forschungsfrage implementiert. Dieser bestätigt, dass eine Wav-Datei mit perkussiven und melodischen (richtig?) Instrumenten in die jeweiligen Gruppen trennbar ist. Der Programmcode und das Ergebnis einer erfolgreichen Durchführung auf eine Wav-Datei werden an diese Arbeit angehängt (wie?). Zudem werden eingangs gestellte Fragen beantwortet und die Hypothesen behandelt.

### **Beantwortung von Unterfragen**

In Abschnitt 3.1 wird erläutert wie ein Ton entsteht und dessen Darstellung in Wellenform aussieht. Anhand der Anzahl an Wellen pro Sekunde wird die Frequenz in Hertz berechnet. Bei der Analyse mehrerer Töne wird es schwieriger einzelne Signale aus einer Wellenform abzulesen (siehe: Abbildung 3.2).

In diesem Projekt werden ausschließlich akustische Instrumente verwendet, da diese aufgrund der natürlichen Entstehung effektiver von der Fourier Transform zugeordnet werden. Dabei werden lediglich perkussive von harmonische Instrumente (jeweils Teil der akustischen Instrumente) getrennt, da deren unterschiedliche Ausschläge

differenzierbar sind. Allerdings konnte im Rahmen dieses Projekts die Wahl der Musikinstrumentgruppen nicht bewiesen, sondern lediglich recherchiert und referenziert werden.

Es wurden WAV, MP3, WMA, AAC, OGG, FLAC und RM als verbreitete Audio-Speicherformen herausgearbeitet. Dabei wurden Wav-Dateien als geeignete Speicherform für dieses Projekt ermittelt, aufgrund des geringen Informationsverlusts. Ebenso hat sich die Fourier Transform als präzisere Transformation im Gegensatz zur Wavelet Transform herausgestellt. Während des Projekts wurde die Short-Time Fourier Transform (STFT) verwendet, die auf der Fourier Transform aufbaut.

## **6.2 Kritische Reflexion der eigenen Arbeit**

Bei der Implementierung des Codes liegt der Fokus auf der Erläuterung von Zusammenhängen, nicht auf der detaillierten Beschreibung einzelner Befehle. Um den Algorithmus nachzuvollziehen, werden Vorkenntnisse im Programmieren benötigt. Für die weitere Arbeit am Programmcode sind zusätzliche Python-Kenntnisse erforderlich (oder?). Allerdings wird für ein Verständnis der behandelten Theorie kein Vorwissen benötigt.

Es werden lediglich vorgegebene Gruppen von Musikinstrumenten getrennt. Die Trennung einzelner Musikinstrumente oder anderer Instrumentengruppen kann der Algorithmus nicht durchführen (oder?). Damit erfüllt der Algorithmus nur einen speziellen Anwendungsfall von den möglichen Instrumenten die getrennt werden.

Das Ergebnis des Projekts ermöglicht ausschließlich die Verwendung von Wav-Dateien. Diese werden aufgrund des hohen Speicherbedarfs selten verwendet und sind schwierig zu erhalten. Bei der Anwendung des Algorithmus auf ein gewähltes Lied, ist es schwierig dieses als Wav-Datei zu finden.

Zudem basiert die Wahl der trennbaren Musikinstrumentgruppen und verwendeten Speicherform lediglich auf einer eingangs gestellten Hypothese. Diese Hypothese kann mittels Quellen und erläuterter Zusammenhänge bekräftigt werden. Allerdings werden diese Thesen nicht durch eine Umsetzung möglicher Alternativen bewiesen.

## **6.3 Ausblick für die Zukunft**

In der Zukunft bieten sich vielfältige Möglichkeiten zur Weiterentwicklung dieses oder ähnlicher Algorithmen, insbesondere im Hinblick auf komplexere Anwendungsfälle. Ein bedeutender Fortschritt wäre die Trennung einzelner Musikinstrumente, anstatt

lediglich harmonische und perkussive Frequenzen zu unterscheiden. So könnten beispielsweise Instrumente wie Bass, Gitarre oder Klavier sowie Schlagzeug und Cajón isoliert werden. Auch komplexere Klangstrukturen, wie gelayerte Synthesizer, könnten mit optimierten Algorithmen analysiert und aufgelöst werden.

Ein solches System würde Musiker:innen enorme Vorteile bieten. Sie könnten die einzelnen Noten ihres Instruments gezielt herausfiltern und zum Üben oder Lernen verwenden, ohne zusätzliche Kosten für Notenmaterial oder separate Audiotracks. Zudem könnte es möglich werden, nicht transkribierte Musikstücke automatisch zu analysieren und in Notenform zu übertragen. Ein weiteres Anwendungsbeispiel wäre die Erstellung von Playbacks: Ein Schlagzeugtrack könnte beispielsweise entfernt werden, um darauf basierend eine eigene Aufnahme zu erstellen.

Der bestehende Algorithmus könnte zudem durch die wiederholte Anwendung mit angepassten Filtern auf die bereits getrennten WAV-Dateien verfeinert werden, um noch spezifischere Ergebnisse zu erzielen. Darüber hinaus bieten moderne Ansätze mit künstlicher Intelligenz eine vielversprechende Möglichkeit, die Erfolgsquote bei der Trennung und Analyse weiter zu erhöhen.

Eine Erweiterung des Algorithmus auf andere Audioformate wie MP3 wäre ebenfalls sinnvoll, um die Ergebnisse auch auf verbreitetere und leichter zugängliche Dateien anzuwenden. Trotz des potenziellen Qualitätsverlusts könnte dies die Nutzbarkeit des Systems für eine breitere Zielgruppe steigern. Ein abschließender Vergleich der Ergebnisse zwischen WAV- und MP3-Dateien würde zusätzlich wertvolle Erkenntnisse über die Leistungsfähigkeit des Algorithmus liefern.

Langfristig eröffnet diese Forschung Perspektiven, um neue Technologien für Musiker:innen, Produzent:innen und Musikliebhaber:innen zugänglicher zu machen, während gleichzeitig der Grundstein für weiterführende Innovationen gelegt wird.

## **Anhang**

Zugehöriges Github Repository