SOUND SEPARATION

Implementierung der Nonnegativen Matrixfaktorisierung zur Signaltrennung von Audiodateien

Zulassungsarbeit

Autorin: Claudia Stockinger
stocki24@gw.uni-passau.de
MatNr. 71762

Studiengang: Lehramt Gymnasium Informatik/Mathematik

Vorgelegt am: 30. März 2020

Betreuer: Prof. Dr. Tomas Sauer

SOUND SEPARATION

Implementierung der Nonnegativen Matrixfaktorisierung zur Signaltrennung von Audiodateien

Bachelorarbeit

Autorin: Claudia Stockinger
stocki24@gw.uni-passau.de
MatNr. 71762

Studiengang: Bachelor Mathematik

Vorgelegt am: 30. März 2020

Betreuer: Prof. Dr. Tomas Sauer

Zusammenfassung

Die vorliegende Arbeit wurde als Bachelorarbeit für Mathematik mit Wahlfach Informatik und als Zulassungsarbeit für das Gymnasiallehramt in Mathematik und Informatik verfasst. Ziel der Arbeit ist es, einen bestimmten Signaltrennungsalgorithmus zu verstehen und zu implementieren, um ihn dann zu testen, mit ähnlichen Algorithmen zu vergleichen und die Qualität der Trennungsergebnisse zu untersuchen.
Dieser Algorithmus basiert auf Nonnegativer Matrixfaktorisierung und integriert ein Gradientenverfahren um zeitliche Kontinuität und Dünnbesetztheit der getrennten Signale zusätzlich zu erreichen. Mit diesem soll eine Trennung einer Audiodatei in unterschiedliche Komponenten ermöglicht werden. Die Basis der Arbeit bildet eine Untersuchung der naturwissenschaftlichen, informationstechnischen und mathematischen Grundlagen, die für ein tiefgreifendes Verständnis dieses Algorithmus erforderlich sind. Der implementierte Algorithmus wird daraufhin genau beschrieben, wobei auch eine kurze Ausführung über die Implementierung selbst folgt.
Der implementierte Code und die getesteten sowie aus den Tests hervorgegangenen Audiodateien werden der Arbeit auf einem Datenträger beigelegt. Das Verfahren wird sowohl mit objektiven als auch subjektiven Maßstäben evaluiert und mit zwei weiteren Verfahren verglichen.

Inhaltsverzeichnis

1 Einleitung

In Anerkennung an die grandiose Leistung aller Musiker, die ihr Talent darauf verwenden, Klänge von analogen oder digitalen Instrumenten in einer kreativen Art und Weise so zusammenzuführen, dass sich dabei Werke ergeben, die der Menschheit oft eine sehr große Freude bereiten, widmet sich diese Arbeit dem Gegenteil dessen – nämlich der Zersetzung derjenigen.
Damit ist nicht die Zersetzung der Musiker gemeint, die die Grundlage des folgenden geistlosen Wortwitzes schafft:

„What is Beethoven doing now?“ De-Composing.

Es handelt sich vielmehr um die *Zersetzung der Werke*. Doch wie können musikalische Werke zersetzt werden und was entsteht dabei? Ohne es bewusst zu merken, zersetzen wir Musik beziehungsweise Geräusche bereits mit unserem Gehör und Gehirn. Beispielsweise erkennen wir in Musikstücken den Takt, die unterschiedlichen Instrumente, die mitwirken und dass die Instrumente für sich gesehen unterschiedliche Töne abspielen.
Es wird viel zur automatischen Musikerkennung und -verarbeitung geforscht – den menschlichen Hörapparat kann die Informationstechnik aber bei Weitem noch nicht ersetzen. Sie verwendet aus der Bildverarbeitung imitierte Signaltrennungsansätze und beschränkt sich im Wesentlichen auf zwei Hauptdisziplinen: „Voice and Music Separation“ und „Spatial Sound Source Separation“. Letztere ist die Zersetzung von Geräuschen in ihre räumlich gesehenen Quellen, das heißt mithilfe von „Spatial Sound Source Separation“ kann man herausfinden, aus welcher Richtung man etwas hört oder wie die Mikrofone bei der Aufnahme gestanden sind. Die „Voice and Music Separation“ befasst sich unter anderem damit, wie man ein ein Instrument aus einem musikalischen Werk extrahiert oder wie man eine Stimme aus einer Geräuschkulisse herausfiltert. Damit ist geklärt, wie man Werke zersetzen könnte.
Die Frage, was dabei entstehen soll, gestaltet sich komplizierter: Die Komponenten eines zerlegten Signals können Verschiedenes repräsentieren. Das reicht beispielsweise bei einem Musikstück von einer feinen Zerteilung einzelner Töne einzelner Instrumente bis hin zu grober Aufteilung in harmonisch und nicht-harmonisch.
In dieser Arbeit sollen drei verschiedene Algorithmen zur Signaltrennung auf einer Audiodatei mit Musikinstrumenten getestet werden. Dabei soll sich eine Trennung in unterschiedliche Instrumente ergeben. Da Musik ein sehr interdisziplinärer Gegenstand ist, können eventuell auftretende Probleme bei einer Tontrennung in naturwissenschaftlichen, mathematischen oder informationstechnischen Hintergründen ihren Ursprung haben. Deshalb ist die Kenntnis dieser Fachrichtungen der Akustik für diese Arbeit essenziell.

2 Naturwissenschaftliche und informationstechnische Grundlagen

Laut Duden ist der Ton eine „vom Gehör wahrgenommene gleichmäßige Schwingung der Luft, die [...] keine Obertöne aufweist“ und die Musik die „Kunst, Töne in bestimmter (geschichtlich bedingter) Gesetzmäßigkeit hinsichtlich Rhythmus, Melodie und Harmonie zu einer Gruppe von Klängen und zu einer stilistisch eigenständigen Komposition zu ordnen“. In ihrem Kern ist Musik also angewandte Mathematik. Im Folgenden werden die physikalisch messbaren Größen zur Beschreibung von Musik erläutert. In diesen Größen haben auch die subjektiv wahrgenommenen Eigenschaften von Musik, wie laut, leise, hoch und tief ihren Ursprung. [Dud]

2.1 Die Schallwelle

Bevor sich Mathematik auf Geräusche anwenden lässt, stellt sich zunächst die Frage, warum sie meistens als Wellen dargestellt werden. Ein Geräusch entsteht durch eine Schallwelle, die eine Schwingung ist, die sich in einem Medium fortpflanzt. Schwingen heißt, sich mit einer gewissen Regelmäßigkeit hin und her zu bewegen. Dabei ist der Begriff Welle etwas irreführend, da eine Welle eigentlich periodisch ist, das heißt sich regelmäßig wiederholt. Schall ist aber nicht immer periodisch, z.B. beim Rauschen. Stellt Luft das Medium dar, schwingen die Luftmoleküle und dies führt zu Druckunterschieden. Dieser Druck erreicht das menschliche Trommelfell, der dadurch ausgelöste Reiz wird im Gehör verarbeitet und an das Gehirn weitergeleitet, sodass diese Druckunterschiede dann ein Hörempfinden in uns auslösen.
Im Folgenden betrachten wir einen einfachen Fall, der sich auf kompliziertere Schallwellen übertragen lässt. Gegeben sei eine ebene Schallwelle, die nur aus einer einzigen homogenen, harmonischen und unendlichen Geräuschquelle entspringt. Eben heißt, dass die Ausbreitungsrichtung der Welle räumlich konstant ist. Die Auslenkung der Luftmoleküle kann durch folgende Differenzialgleichung beschrieben werden, die allgemein als Wellengleichung bezeichnet wird:

$$\frac{\partial^2 \zeta}{\partial t^2} = c^2 \frac{\partial^2 \zeta}{\partial x^2} \tag{1}$$

Hierbei bewegt sich das Luftmolekül, das ursprünglich an der Position x war, um ζ in der Zeit t. Im Folgenden beschreibt ζ_m die maximale Auslenkung, also die sogenannte Amplitude der Schwingung und λ ist der kleinste Abstand zweier Punkte mit der gleichen Auslenkung, auch als Wellenlänge bezeichnet. Außerdem ist c eine Variable, die durch die Dichte und Elastizität des Mediums, z.B. der Luft, zustande kommt. ζ_m und λ sind Konstanten, also ist eine Lösung der Differenzialgleichung (1) für alle

Anfangswerte:

$$\zeta(t,x) = \zeta_m cos 2\pi(\frac{c}{\lambda}t \pm \frac{x}{\lambda}) \tag{2}$$

Der Beweis dazu ist trivial: Es wird (2) in (1) eingesetzt und damit deutlich, dass die Gleichheit gegeben ist. Die Randbedingungen dieser Gleichung ergeben sich aus der Beschaffenheit des Ausbreitungsraumes, deshalb ist es wenig interessant, sie an dieser Stelle mathematisch zu untersuchen. Bei dieser Gleichung kann man entweder t oder x konstant halten. Hält man t konstant, so beschreibt die Gleichung den räumlichen Verlauf der Schwingung, also die Differenz der Moleküle zur Ruhelage zur Zeit t. Hält man x konstant, so ist die Gleichung ein Ausdruck des zeitlichen Verlaufs der Position eines einzigen Luftmoleküls. Für beide Fälle ist die Funktion eine sinusförmige Funktion und ist auf ganz $\mathbb{R}$ definiert, da der Abstand λ zweier Moleküle stets $\lambda \neq 0$ erfüllt. Sie ist periodisch mit Periode λ, das heißt es gilt

$$\zeta(t) = \zeta(t+\lambda) \text{ bzw } \zeta(x) = \zeta(x+\lambda)$$

Obwohl Schallwellen in der Luft Longitudinalwellen sind, werden Töne meist als Transversalwellen dargestellt, da man diese einfacher als Funktion umschreiben und leichter interpretieren kann. Longitudinalwellen sind Wellen, die längs der Ausbreitungsrichtung schwingen. Transversalwellen sind Wellen, deren Schwingung senkrecht zur Ausbreitungsrichtung erfolgt. Abbildung 1 visualisiert einen Vergleich der beiden Darstellungen und zeigt, dass sich die Informationen einer Longitudinalwelle gut in einer Transversalwelle darstellen lassen. Die Kreise symbolisieren die Luftmoleküle im Ruhezustand. Dabei bedeutet die Auslenkung nach oben und unten in der Transversalwelle die Auslenkung aus der Ruhelage nach links und rechts in der eigentlichen Longitudinalwelle. Die Darstellung eines Geräusches als Transversalwelle wird Oszillogramm genannt. [Ran05, S.42-45]

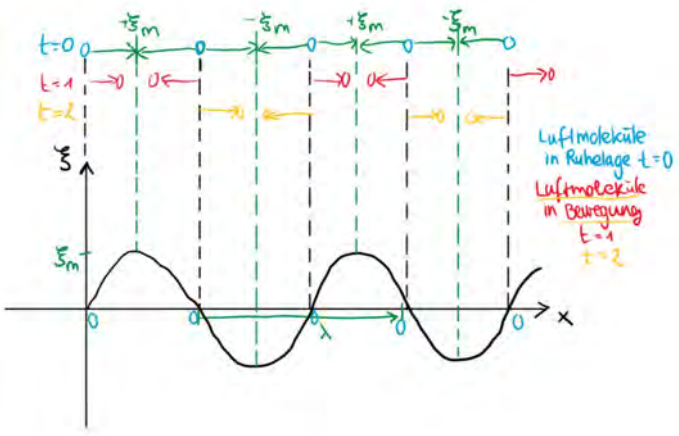

Abbildung 1: oben Longitudinalwelle, unten Transversalwelle einer Sinusschwingung

2.2 Physikalische Größen

Die geläufigste Maßeinheit für die Tonhöhe ist Hertz, für die Lautstärke des Tons Dezibel. Außerdem nimmt der Mensch neben Lautstärke und Tonhöhe auch die Klangfarbe einer Schallwelle wahr, jedoch gibt es dafür keine Maßeinheit. Betrachten wir die Funktion (2) als zeitabhängige Funktion, also:

$$\zeta(t) = \zeta_m cos 2\pi(\frac{c}{\lambda}t \pm \frac{x}{\lambda})$$

Die Frequenz F der Welle wird in Hertz (Hz) angegeben und ist der Kehrwert der Wellenlänge, also $F = \frac{1}{\lambda}Hz$. Dabei ist die Wellenlänge die Zeit, die vergeht, bis sich ein Wellenbild wiederholt. Je höher die Frequenz ist, desto höher nimmt der Mensch den Ton wahr, wobei er nur Töne hören kann, die circa zwischen 20 und 20000 Hz liegen. Natürlich gibt es Funktionen mit der gleichen Frequenz aber mit unterschiedlichem Bild, das heißt unterschiedlicher Wellenform. Das sind Funktionen von Tönen mit gleicher Höhe aber unterschiedlicher Klangfarbe.
Die Lautstärke L ist abhängig von der Schallintensität $I = p \cdot v$, die auf das Trommelfell auftrifft, und kann in Watt pro Quadratmeter ($\frac{W}{m^2}$) gemessen werden. Dabei ist $v = \frac{\partial \zeta}{\partial t}$ die Schallschnelle und p der Schalldruck, die auf eine gewisse Fläche wirkende Kraft des Schalls. Die Schallschnelle ist je nach Bedingungen des Ausbreitungsmediums konstant und beträgt z.B. in der Luft bei 20 Grad Celsius 343 m/s. $I_0 = 10^{-12}\frac{W}{m^2}$ ist die durchschnittliche unterste Hörbarkeitsschwelle, $10^{12}\frac{W}{m^2}$ die durchschnittliche oberste Schmerzgrenze. Da diese Größen relativ unhandlich sind, wird für die Lautstärke L eine andere Maßeinheit benutzt, und zwar Dezibel (dB).

$$L = 10 log_1 0 \frac{I}{I_0}\ dB$$

Sie gibt an, wie die Schallintensität im Verhältnis zur untersten Hörschwelle liegt. Außerdem ist sie logarithmisch skaliert, denn die Wahrnehmung der Lautstärke durch den Menschen folgt logarithmischen Prinzipien: Wird die Schallintensität verdoppelt, so wird die Lautstärke nicht als doppelt so laut empfunden, sondern eigentlich nur als geringfügig lauter. Diese Umformung führt zu den handlichen Größen 0 dB als unterste Hörschwelle und 120 dB als oberste Schmerzgrenze. [Ack91, S.8f]

2.3 Töne, Klänge, Geräusche

Es gibt verschiedene Bezeichnungen für die Empfindung von Schallwellen, welche auf der Form der Welle basieren. Es wird zum Beispiel zwischen Ton, Klang und Geräusch differenziert.
Geräusch bezeichnet alle Empfindungen, die nicht als Ton(-gemisch) oder Klang(-gemisch) kategorisiert werden können, weil die Funktion (2) nicht periodisch ver-

läuft. Dazu gehört auch unter anderem Rauschen.
Als Ton empfinden wir eine Schallwelle, wenn deren Funktion eine trigonometrische Funktion ist, so wie z.B. y_1 in nachfolgender Abbildung 2. Ein reiner Ton kommt in der Natur allerdings kaum vor – er wird normalerweise überlagert von mehreren Obertönen. Überlagern heißt, dass die Funktionen der Töne addiert werden. Falls die Obertöne die Harmonischen sind, dann sind sie Töne, die als Frequenz ganzzahlige Mehrfache von der Grundfrequenz des Tons besitzen, wie z.B. y_2, y_3 *und* y_4 in der Abbildung 2. Überlagern sich diese Wellen, ergibt sich dann ein als harmonisch empfundener Klang, ein Beispiel hierfür ist y_{14} in der Abbildung 2. Ist das Verhältnis von Ton und Oberton nicht genau ganzzahlig, so wird der Klang als unharmonisch wahrgenommen.
Der Klang y_{14} lässt sich alternativ in einem Amplituden-Frequenzdiagramm abbilden, auch als Fourierspektrum bezeichnet. Darin wird für den Grundton und die Obertöne einzeln zur jeweiligen Frequenz die Amplitudengröße eingetragen, so wie in Abbildung 2 rechts.

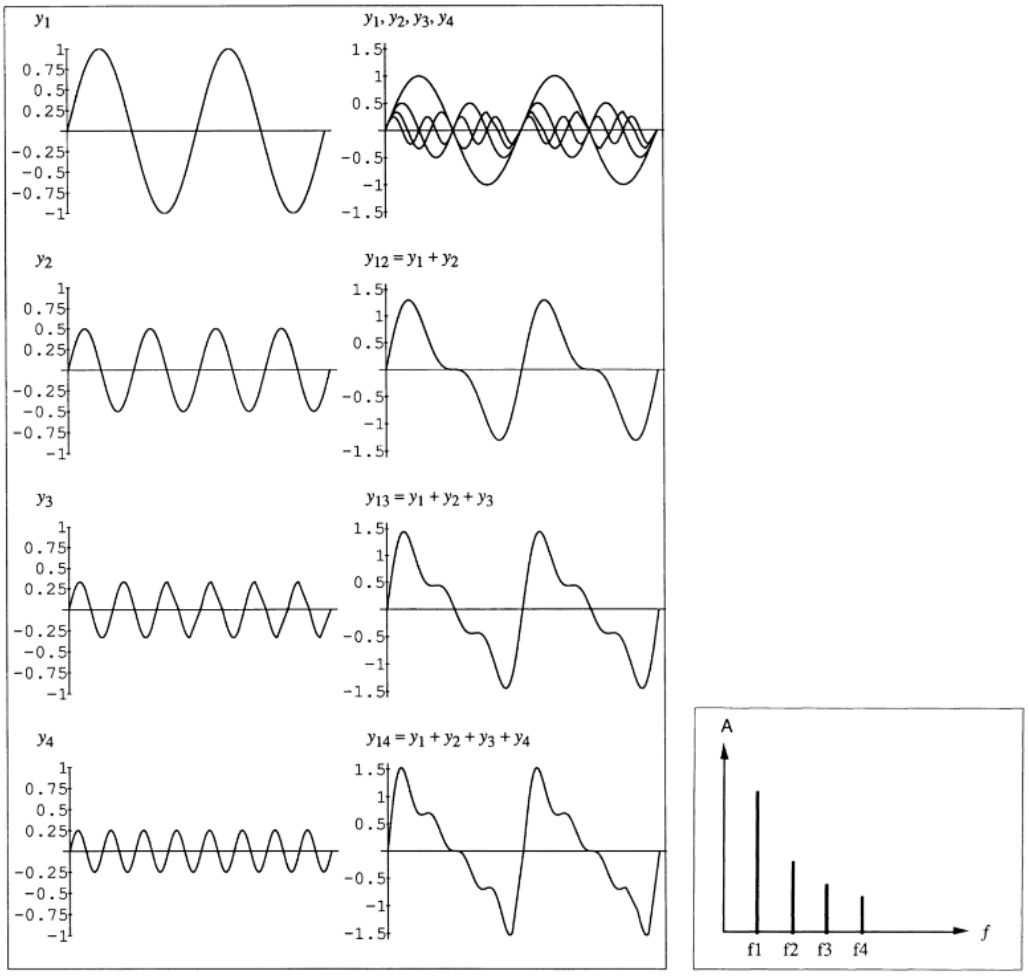

Abbildung 2: Links das Oszillogramm, rechts das Fourierspektrum des Klangs y_{14}

Es gilt, dass die Empfindung der Klangfarbe allein von den Amplituden der einzelnen Töne des Klangs abhängt, und nicht von deren Auslenkung der Schwingung, welche Phase genannt wird. Sind die einzelnen Töne die gleichen, die Phasen derer jedoch verschieden, so können sich mehrere verschiedene Oszillogramme ergeben, die alle den gleichen Klang repräsentieren. Ein Vorteil des Fourierspektrums gegenüber dem Oszillogramm, ist, dass hierbei die Phase keine Rolle spielt und die Darstellung deshalb eindeutig ist. Betrachtet man z.B. die beiden Töne $y_5 = sin(2\pi\frac{t}{\lambda})$ und $y_6 = sin(2\pi\frac{t}{\lambda} - \frac{\lambda}{2})$, so haben diese zur Zeit t je eine unterschiedliche Phase und somit insgesamt ein unterschiedliches Oszillogramm, aber dennoch ein gleiches Fourierspektrum. Beide Töne besitzen die gleiche Amplitude und Wellenlänge und werden deshalb als exakt derselbe Ton wahrgenommen. [Ran05, S.14 f]

2.4 Digitalisierung der Schallwelle

Mithilfe eines Mikrofons kann eine Schallwelle in elektrische Spannung umgewandelt und dadurch gespeichert und manipuliert werden. Sowohl der Schall als auch die elektrische Spannung sind zeitkontinuierlich. Beide werden auch analoge Signale genannt. Würde man die elektrische Spannung exakt abspeichern wollen, müsste man unendlich viele Werte speichern, da es wegen der Kontinuität allein schon in der ersten Sekunde unendlich viele Zeitpunkte mit unterschiedlichen Spannungswerten gibt. Deshalb akzeptiert ein Computer nur diskrete und nicht kontinuierliche Werte. Dazu wird die Welle abgetastet und der Computer speichert sie als eine Abfolge von Frequenzzuständen zu beliebig kleinen Zeitabständen. Dieser Vorgang nennt sich Digitalisierung. Das zeitliche Intervall zwischen zwei Abtastungen bestimmt die Abtastfrequenz der Digitalisierung. Bei einer Abtastrate von 44100 Hz wird 44100 mal pro Sekunde die Größe der Schwingung gemessen. Die Abtastauflösung bestimmt, mit welcher Genauigkeit die Zustandswerte gespeichert werden. Beispielsweise bedeutet die Abtastauflösung 8 Bit, dass der Messwert durch $2^8 = 256$ Nullen und Einsen in Binärform dargestellt werden kann. Die gespeicherten diskreten Werte können dann unter Hinzunahme der Abtastauflösung und der Abtastfrequenz wieder in analoge Signale verwandelt werden.

Definition 1. *Ein Signal ist eine Funktion $x : K \to \mathbb{C}$ über t, wobei t die Zeit beschreibt.*

Analoge Signale sind stückweise stetig mit $K \subseteq \mathbb{C}$, wohingegen bei digitalen Signalen K eine abzählbare Menge der Kardinalität N ist und die Funktion x somit diskret. N berechnet sich aus der Abtastfrequenz mal der zeitlichen Länge des Signals. Die Funktion ist grundsätzlich nicht zeitbeschränkt, also $t \in \mathbb{R}$, aber wenn das Signal am Computer gespeichert wird, muss man von einer Zeitbeschränktheit ausgehen, da Aufnahmen natürlich nicht von unendlicher Zeit sein können.

Mathematisch funktioniert die Umwandlung von analogen $x_{analog}(t)$ zu digitalen Signalen $x_{digital}(t)$ relativ einfach, indem man die analoge Funktion zuerst mit einer periodischen Pulsfunktion $s(t)$ multipliziert, also $x(t) = s(t) \cdot x_{analog}(t)$. Ein Beispiel für eine periodische Pulsfunktion $s(t)$ ist die charakteristische Funktion. Sei T die Menge der Abtastzeitpunkte, dann gilt für sie:

$$s(t) = \mathbf{1}_T = \begin{cases} 1, & \text{falls } t \in T, \\ 0, & \text{falls } t \notin T \end{cases}$$

Zuletzt muss der Wert noch an die Abtastauflösung angepasst werden. Sei A die Abtastauflösung. Die Umwandlung erfolgt durch:

$$x_{digital} = A\lfloor \frac{x(t)}{A} \rfloor$$

Bei digitalen Signalen wird zwischen Mono-, Stereo- oder Surround-Sound unterschieden. Bei Mono-Sound ist das Signal ein Vektor, bei Stereo- oder Surround-Sound wird das Signal in zwei oder mehrere unterschiedliche Vektoren aufgeteilt, und je ein Signalvektor über einen Kanal zu einem Lautsprecher übertragen. Bei zwei oder mehreren Lautsprechern kann man damit z.B. eine Schallwelle räumlich wahrnehmbar machen, das heißt man kann die Illusion erzeugen, dass die Schallwelle aus unterschiedlichen Richtungen kommt. [Ack91, S.76f] [Woy19, S.3f]

2.5 Fehlerquellen bei der Digitalisierung

2.5.1 Aliasing

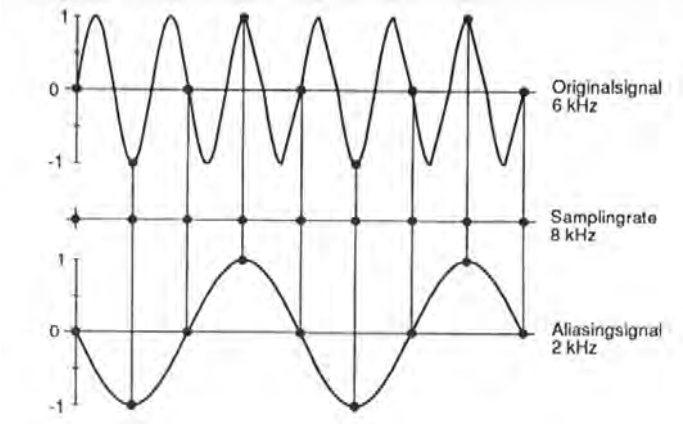

Abbildung 3: Beispiel für Aliasing

Wie in Abbildung 3 erkennbar ist, kann eine zu geringe Abtastfrequenz, auch Samplingrate genannt, dazu führen, dass ein Signal nicht eindeutig zurückverwandelt werden kann. Um dies zu vermeiden, sollte man gemäß dem Nquist-Shannon

Theorem als Abtastfrequenz immer eine Abtastraste verwenden, die mindestens doppelt so groß wie die größte vorkommende Frequenz des abzutastenden Signals ist. Deshalb ist z.B. der Standard auf CD-ROMs eine Abtastrate von 44100 Hz, denn, wie bereits erwähnt, reicht das menschliche Gehör bis circa 20000 Hz. Es können zwar Frequenzen über 20000 Hz im Signal vorkommen, aber alles was über dieser Frequenz liegt muss nicht mehr gespeichert werden, weil der Mensch dies ohnehin nicht hören könnte. [Ack91, S.82]

2.5.2 Filterung

Filtern bedeutet, das Signal nach bestimmten Kriterien zuzuschneiden, also beispielsweise eine Zeitsequenz oder ein Spektrum an Frequenzen abzuziehen. Will man Aliasing vermeiden, so muss man vor der Digitalisierung durch einen Filter alle Frequenzen abschneiden, die über der Hälfte der Abtastfrequenz liegen. Dies kann bei zu niedriger Abtastfrequenz – vor allem wenn sie unter dem Standard von 44100 Hz liegt – wiederum zu Verzerrungen und Phasenverschiebungen führen. [Ack91, S.81]

2.5.3 Quantisierungsrauschen

Die Begrenzung durch die Abtastauflösung führt unvermeidlich dazu, dass die Amplitude des Signals nicht immer richtig und exakt abgebildet werden kann. Diese eher kleinen Fehler sind bei der Verarbeitung nicht von großer Bedeutung, können aber bei der Nutzung, also z.B. bei der Wiedergabe des Signals, eventuell als Summen wahrgenommen werden. Das Summen kann reduziert werden, indem zu dem analogen Eingangssignal vor der Digitalisierung ein Rauschen mit geringer Amplitude addiert wird. [Ack91, S.82]

2.6 Maschinelles Lernen

Ziel dieser Arbeit ist es, ein Signal in seine einzelnen Komponenten mithilfe eines Algorithmus zu zerlegen. Selbst der Mensch ist nicht immer dazu fähig, alle Komponenten – insbesondere Instrumente – eines Musikstücks heraus zu hören, also Klangmuster zu erkennen. Dazu braucht er ein geübtes Gehör, aber vor allem kann der Mensch kaum feste Regeln zur exakten Erkennung definieren. Er kann höchstens Eigenschaften beziehungsweise Muster beschreiben, die einzelne Komponenten aufweisen sollten. Ähnlich dazu heißt das für die Informationstechnik, dass man zur Signaltrennung einen dynamischen Ansatz verwenden muss, der dem menschlichen Lernen und Analysieren ähnlich ist. Seit es Computer gibt, ist auch von maschinellem Lernen die Rede, wobei das Spektrum des Erreichbaren eng mit der Rechenleistung der Computer zusammenhängt. Maschinelles Lernen ist die künstliche Erzeugung von Wissen aus existierenden Daten. Methoden des maschinellen Lernens werden

angewendet, um Daten zu assoziieren, klassifizieren und Vorhersagen zu treffen. Man kann zwei Hauptrichtungen von maschinellem Lernen unterscheiden [Ala08, S.11]:

- **überwachtes Lernen**: Hier ist das Ziel, eine Abbildung zu finden, die bestimmte Eingabewerte auf bestimmte Ausgabewerte abbildet. Dabei ist bekannt, welche Ausgabewerte korrekt sind. Man erreicht dies, indem man die Parameter der Abbildung in einem Optimierungsverfahren immer wieder aktualisiert bis die Abbildung die gewünschten Ergebnisse liefert.

- **unüberwachtes Lernen**: Bei unüberwachten Lernen hat man nur die Eingabedaten, die korrekten Ausgabewerte stehen einem nicht zur Verfügung. Das Ziel besteht dann eher darin, Muster und Regelmäßigkeiten in den Eingabedaten zu finden.

Schon allein aus dem Grund, dass man die Anzahl der Komponenten des Eingabesignals nicht algorithmisch herausfinden kann, muss man auf unüberwachte Lernalgorithmen zurückgreifen. Eine weitere Entscheidung, die man zur Auswahl des Algorithmus treffen muss, ist, was man genau trennen möchte – einzelne Instrumente, einzelne Noten, das Tempo, etc. Es ist komplizierter, jede einzelne Quelle in einem Signal exakt zu erkennen, als das Signal nur in harmonische und nicht harmonische Anteile zu trennen. Sei n die Anzahl der ursprünglichen Quellkomponenten des zu trennenden Signals und m die Anzahl der Ergebniskomponenten nach der Signaltrennung. Es können folgende drei Fälle eintreten:

- n = m. Dies ist der derterminierte Fall. Man muss hier nur noch die Ergebniskomponenten den Quellkomponenten zuordnen.

- n < m. Dies ist der überdeterminierte Fall. Eventuell wurde hier eine Quellkomponente zweigeteilt und der Fehler lässt sich leicht durch Addieren der Komponenten beheben.

- n > m. Der unterdeterminierte Fall ist am problematischsten; er erfordert eine erneute Signaltrennung für ein korrektes Ergebnis.

Seien $S = [s_1, ..., s_n]$ die n Quellkomponenten der Länge t, s ist also von der Dimension $n \times t$. Sei X das zu trennende Signal der Länge t. Dann entstand X ursprünglich dadurch, dass S mit einer Mischmatrix A multipliziert wurde, also $X = AS$.
Um beim überwachten Lernen die optimale Abbildung des zu trennenden Eingabesignals auf seine im Vorhinein bekannten Quellkomponenten zu finden, so heißt es, eine Mischmatrix A zu finden, sodass $A = XS^{-1}$.
Beim unüberwachten Lernen ist die Mischmatrix A ebenfalls zu finden - aber weil man die richtigen Ergebnisse nicht kennt - so, dass sie für $S = A^{-1}X$ Ergebnisse

erzeugt, die sinnvolle Signalkomponenten erahnen lassen. Diese Mischmatrix wird durch die Kriterien, mit denen man Muster und Regelmäßigkeiten zu erkennen versucht, erzeugt. Ansätze, bei denen die Mustererkennung für Audiosignale in die Praxis umgesetzt wurde, sind zum Beispiel die Independent Component Analysis, Clustering oder Nonnegative Matrixfaktorisierung. In Kapitel 3 wird auf die mathematischen Grundlagen der Nonnegativen Matrixfaktorisierung genauer eingegangen. [MEKR11]

2.7 Mustererkennung in Audiosignalen

Signaltrennung kann auch durch Aufteilen in Frequenzabschnitte geschehen, wobei aber selten ein Ergebnis mit sinnvollen Signalkomponenten erzeugt wird. Deshalb sollte man zur Auswahl der Trennungsmethode vorher eine Entscheidung treffen, wie ein korrektes Ergebnis aussieht. Dabei kann zurate gezogen werden, dass Audiosignale bestimmte Eigenschaften haben und die Ergebnisse diese Eigenschaften auch bestmöglich aufweisen sollten.
Sei x das zu trennende Signal und seien $x_1, ..., x_n$ Signalkomponenten, die aus der Signaltrennung hervorgehen. Über die ursprüngliche Quellkomponenten gibt es keine Informationen vorab – weder wie viele es sind, noch die Frequenzbereiche der einzelnen Komponenten sind bekannt. Folgende Eigenschaften können zur Mustererkennung der Signalkomponenten festgelegt werden:

2.7.1 Stochastische Unabhängigkeit und Additivität

Wurde ein Klang durch das Spielen eines harmonischen Instruments erzeugt, so sind der Grundton und die Obertöne nicht stochastisch unabhängig voneinander, denn immer wenn der Grundton auftritt, treten auch die Obertöne auf. Zwei Ereignisse A und B (z.B. A=„Der Grundton tritt auf“ B=„Die Obertöne dieses Grundtons treten auf“) sind stochastisch unabhängig voneinander, wenn die Wahrscheinlichkeit, dass beide Ereignisse eintreten, gleich dem Produkt ihrer Einzelwahrscheinlichkeiten ist, also $P(A \cap B) = P(A) \cdot P(B)$. Wird dieser harmonische Klang dann in zwei Signalkomponenten aufgeteilt, z.B. eine Komponente für den Grundton und eine Komponente für die Obertöne, so sind beide Komponenten stochastisch abhängig voneinander und man erkennt daran, dass ein Fehler in der Trennung unterlaufen ist. Außerdem kann man von einer korrekten Signaltrennung verlangen, dass die Signalkomponten gemeinsam abgespielt wieder den Klang des ganzen Signals ergeben, also $x = x_1 + ... + x_n$. [Vir07]

2.7.2 Dünnbesetztheit

Die Komponenten sollten dünnbesetzt sein, das heißt der Wert der Funktion (2) der Signalkomponente ist an wenigen Stellen ungleich 0. In einem Musikstück gibt es

selten einen gleichbleibenden Klang, der über die Dauer des Stückes durchgängig gespielt wird. Eher ist es so, dass viele verschiedene Instrumente immer wieder unterschiedliche Töne anstoßen, somit ein einzelner Ton eines einzelnen Instrumentes eher sehr selten auftritt und dadurch das Signal, dass diesen Ton beschreibt, dünnbesetzt ist. Ein Gegenbeispiel für eine dünnbesetzte Komponenente kann auch ein durch Rauschen gestörtes Signal sein. [Vir07]

2.7.3 Zeitliche Kontinuität

Stößt ein Instrument einen Ton an, so entsteht im Normalfall eine kontinuierliche Welle, die mit der Zeit abflacht und keine starken plötzlichen Lautstärkeveränderungen aufweist. Eine Diskontinuität könnte in der Aufnahme durch menschliche Manipulation entstehen, z.B. durch Verdecken des Aufnahmegerätes oder Bearbeitung und Zuschnitt der Audiodatei. Ignoriert man diese selteneren Einzelfälle, kann man festlegen, dass für die Funktionen der Signale gilt, dass sie zeitlich kontinuierlich sind. Zeitlich kontinuierlich bedeutet einerseits Stetigkeit im analogen Fall und andererseits, dass für nah aneinander liegende Zeitpunkte t_1 und t_2 des Signals x gilt, dass $x(t_1) - x(t_2)$ einen geringen Wert hat. [Vir07]

3 Mathematische Grundlagen

Da es Ziel dieser Arbeit ist, einen Algorithmus zur Signaltrennung am Rechner zu implementieren, wird in der folgenden Einführung der mathematischen Grundlagen der Fokus auf zeitbeschränkte diskrete Signale gelegt.

3.1 Approximation und Interpolation

Wie schon erwähnt, sind Signale am Computer nur als einzelne unzusammenhängende Werte abgespeichert. Um die Höhe und Lautstärke eines Klangs zu bestimmen und ihn so einer Komponente zuordnen zu können, müssen wir die Wellenlänge und Amplitude des Signals herausfinden. Dies gestaltet sich allerdings bei einer Funktion schwer, bei wegen der Diskretheit diese Werte nicht abgelesen beziehungsweise berechnet werden können. Um sich einer solchen Funktion für das diskrete Signal zu nähern, muss man interpolieren und approximieren.
Interpolation bedeutet zu einer vorgegebenen Anzahl von Punkten – z.B. ein abgetastetes Signal – eine Funktion zu suchen, deren Graph genau durch diese Punkte geht.
Approximation von Funktionen bedeutet in der Mathematik den Versuch, eine Funktion durch einfachere Funktionen bestmöglich anzunähern, z.B. mithilfe der Fouriertransformation. Man erhält mit der Fouriertransformation eine Reihe von Sinusfunktionen, die miteinander addiert die zu approximierende Funktion ergeben.
Interpolation und Approximation sollten nicht verwechselt werden, denn dass man durch korrekte Interpolation gleichzeitig eine gute Approximation findet, ist nicht immer gegeben. Ein gutes Beispiel dafür ist Aliasing (siehe Kapitel 2.5.1).
Aber Interpolation ist auf jeden Fall ein Weg, eine Approximation zu finden. Für digitale Signale kann man eine Approximation mithilfe der Diskreten Fouriertransformation erhalten, die eigentlich eine Interpolation ist.

Gradient als Beispiel für eine Approximation

Definition 2. *Der **Gradient** einer partiell differenzierbaren Funktion $f : \mathbb{R}^n \to \mathbb{R}$ im Punkt $p = (x_1, ..., x_n)$ ist der Vektor der partiellen Ableitungen in diesem Punkt*

$$(\nabla f)(p) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{pmatrix} \tag{3}$$

Man kann den Gradienten auch allgemein, ohne Bezugnahme auf einen bestimmten Punkt berechnen. Dieser Gradient definiert dann eine Funktion: $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$

Definition 3. *Linearisierung*
Die Linearisierung im Eindimensionalen (also n=1 in Definition 2) geschieht wiefolgt:

$$f(x) = f(p) + f'(p)(x - p) + R(x)$$

Ähnlich geschieht die Linearisierung auch im Mehrdimensionalen:

$$f(x) = f(p) + (\nabla f)(p) \cdot (x - p) + R(x)$$

Interessant ist hier das Umstellen der Formel (7), denn dann gilt
$f'(p) = \frac{f(x)-f(p)}{x-p} + R(x)$. Der Erste Summand erinnert an den Differenzquotieten. Der Restterm R(x) erklärt sich dadurch, dass eigentlich
$f'(p) = \lim_{x\to p} \frac{f(x)-f(p)}{x-p} = lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$.
Der Term $f(p) + (\nabla f)(p) \cdot (x - p)$ beschreibt eine Tangente im Punkt x. Eine Tangente an einer Funktion kann als Approximation der Funktion gesehen werden. Die Ableitung f' beziehungsweise der Gradient ∇f ist somit eine lineare Approximation von f.
[AHK+12, S.802f]

3.2 Die Fouriertransformation

Um ein Audiosignal zu analysieren, müssen wir herausfinden, welche Töne mit welcher Frequenz beziehungsweise welche Klänge mit welchen Frequenzanteilen zu bestimmten Zeitpunkten in welchem Ausmaß auftreten. Die Fouriertheorie besagt, dass jede Funktion als Summe trigonometrischer Polynome approximiert werden kann. Da auch ein Ton- und Klanggemisch eine Summe von Sinuswellen ist, ist die Fouriertransformation das perfekte Werkzeug, um einzelne Töne dieser Gemische herauszufiltern.

Definition 4. *Ein* ***trigonometrisches Polynom vom Grad n*** *ist eine Funktion* $p : \mathbb{R} \to \mathbb{C}$ *der Form:*

$$p(x) = \sum_{-n}^{n} c_k e^{ikx}, \quad x \in \mathbb{R}$$

Sei T die Menge der ***trigonomentrischen Polynome***

$$T = \langle \{e^{ikx} \mid k \in \mathbb{Z}\} \rangle = \langle \{1\} \bigcup \{cos(kx), sin(kx) \mid k \in \mathbb{N}\} \rangle \tag{4}$$

Sie besitzt unendliche Dimension, da die Anzahl der Basiselemente unendlich ist. T_n *sind alle trigonometrischen Polynome bis zum Grad n.*

Die Gleichheit in (4) gilt wegen der Euler'schen Formel:
$e^{ix} = cos(x) + i \cdot sin(x), x \in \mathbb{R}$.

Definition 5. *Der Vektorraum $L^2(-\pi,\pi)$ der* ***quadratintegrierbaren Funktionen*** *auf $(-\pi,\pi)$ ist definiert als die Menge*

$$L^2(-\pi,\pi) = \{f : (-\pi,\pi) \to \mathbb{C} \;\mid\; \int_{-\infty}^{\infty} |f(x)|^2 dx < \infty\}$$

Das Besondere am Raum der quadratintegrierbaren Funktionen ist, dass hier zwei Funktionen miteinander identifiziert werden, wenn ihr Abstand im quadratischen Mittel 0 ist. Das heißt, hier wird zum Beispiel auf dem Intervall $(-\pi,\pi)$ eine Sinusfunktion und dieselbe Sinusfunktion bloß mit Phasenverschiebung nicht voneinander unterschieden.

Definition 6. *Das* ***komplexe Fourierpolynom*** *p_n vom Grad n zu einer Funktion $f \in L^2(-\infty,\infty)$ ist definiert als*

$$p_n(x) = \sum_{k=-n}^{n} c_k e^{ikx}, \quad x \in \mathbb{R}$$

mit den ***Fourierkoeffizienten***

$$c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx} dx$$

Es ist das eindeutig bestimmte Polynom aus T_n, welches f im quadratischen Mittel am besten approximiert.

Das Fourierpolynom lässt sich auch im Reellen bilden, indem man die Umformungen $a_0 = c_0,\ a_k = c_k + c_{-k},\ b_k = i(c_k - c_{-k})$ anwendet.

Definition 7. *Das* ***reelle Fourierpolynom*** *p_n vom Grad n zu einer Funktion $f \in L^2(-\infty,\infty)$ lautet*

$$p_n(x) = a_0 + \sum_{k=1}^{n} (a_k cos(kx) + b_k sin(kx)) \tag{5}$$

für $x \in \mathbb{R}$ mit den Koeffizienten

- $a_0 = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x)dx$
- $a_k = \frac{1}{\pi} \int_{-\infty}^{\infty} f(x)cos(kx)dx$
- $b_k = \frac{1}{\pi} \int_{-\infty}^{\infty} f(x)sin(kx)dx$

für k = 1,2,3,...

Definition 8. *Ist $f : D \to \mathbb{R}$ eine relle Funktion, dann gilt*
f ist ***gerade****, genau dann wenn $f(x) = f(-x) \ \ \forall x \in D$*
f ist ***ungerade****, genau dann wenn $f(x) = -f(-x) \ \ \forall x \in D$*

Für periodische Funktionen gilt, dass die Funktion genau dann ungerade ist, wenn $f(0) = 0$. Die Funktion f ist also schon dann ungerade, wenn bei der Schallwelle am Zeitpunkt $t = 0$ von der Ruhelage ausgegangen wird, was meistens der Fall ist. Das Fourierpolynom (5) lässt sich für ungerade Funktionen vereinfacht darstellen, es gilt nämlich:
Ist f eine ungerade Funktion, so ist $a_k = 0$ für $k = 0, 1, 2, ...$ und
$b_k = \frac{2}{\pi} \int_0^\pi f(x)sin(kx)dx$ für $k = 1, 2, 3, ...$

Satz 1. ***Fourier'scher Entwicklungssatz***
Die Folge (p_n) der Fourierpolynome zu einer Funktion $f \in L^2(-\infty, \infty)$ konvergiert im quadratischen Mittel gegen f, d.h.

$$\int_{-\pi}^{\pi} |p_n(x) - f(x)|^2 dx \longrightarrow 0 \quad (n \to \infty)$$

Die in diesem Sinne konvergente Reihe

$$(\sum_{-\infty}^{\infty} c_k e^{ikx})$$

mit den Fourierkoeffizienten (c_k) von f heißt ***Fourierreihe****.*

Ein Beweis der Konvergenz der Fourierreihe findet man in [AHK+12, S.1044]. Dieses Resultat liefert uns in der akustischen Praxis die Errungenschaft, dass wir einen Klang durch Addieren verschiedener Töne beliebig gut erzeugen können.

Definition 9. *Zu einer über $\mathbb{R}$ integrierbaren Funktion f ist die* ***Fouriertransformation*** *definiert durch*

$$\mathcal{F}(f)(s) = \int_{-\infty}^{\infty} e^{-ist} f(t)dt$$

für $s \in \mathbb{R}$. Ihre Umkehrung, die ***inverse Fouriertransformation****, ist auf $f \in L^2(-\infty, \infty)$ durchführbar und ist für $t \in \mathbb{R}$ gegegeben durch:*

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ist}(\mathcal{F}(f)(s))ds$$

Die obig angeführten Sätze gelten für analoge Signale. Auf diskrete Signale kann auch eine Fouriertransformation angewendet werden, sie nennt sich die Diskrete Fouriertransformation. Sie ist eine trigonometrische Interpolation bei der für das Intervall $(-\pi, \pi)$ die äquidistanten Interpolationspunkte x_j durch

$$x_j = -\pi + j\frac{\pi}{N}, \quad j = 0, ..., 2N, \quad N \in \mathbb{N}$$

und die Interpolationsaufgabe durch das Finden eines trigonometrischen Polynoms p zur 2π-periodischen Funtkion $f : [-\pi, \pi] \to \mathbb{C}$ gegeben ist, sodass

$$f(x_j) = p(x_j), \quad j = 0, ..., 2N \quad N \in \mathbb{N}$$

Durch die Anzahl 2N ist sichergestellt, dass die Anzahl der Interpolationspunkte gerade ist. Da f 2π-periodisch ist, gilt $f(x_0) = f(-\pi) = f(\pi) = f(x_{2N})$ und damit ist es hinreichend, die Interpolation für das beschränkte Intervall $[-\pi, \pi]$ zu finden, um die Interpolation auf dem gesamten Definitionsintervall zu bestimmen.

Satz 2. *Zu jeder 2π-periodischen Funktion ϕ gibt es ein eindeutig bestimmtes trigonometrisches Interpolationspolynom beziehungsweise eine eindeutig bestimmte diskrete Fouriertransformation q_n, deren Koeffizienten des Fourierpolynoms*

$$p_n(x) = \sum_{k=-N-1}^{N} c_k e^{ikx}, \quad x \in \mathbb{R}$$

durch

$$c_k = \frac{1}{2N} \sum_{j=0}^{2N-1} \phi(x_j) e^{-ikx_j} \quad k = -N+1, ..., N$$

gegeben sind.

Dieses trigonometrische Interpolationspolynom ist die Diskretisierung des Fourierpolynoms.

Definition 10. *Durch das Bilden von Potenzen der n-ten Einheitswurzel $w_n = exp(\frac{2\pi i}{n})$ erhält man die sogenannte **Fourier-Matrix** W_n*

$$W_n = \begin{pmatrix} w_n^{0\cdot 0} & \cdots & w_n^{0\cdot(n-1)} \\ \vdots & & \vdots \\ w_n^{(n-1)\cdot 0} & \cdots & w_n^{(n-1)\cdot(n-1)} \end{pmatrix}$$

Die Fouriermatrix ist symmetrisch, also $W_n = W_n^T$. Für die komplex konjugierte Matrix gilt außerdem $W_n\overline{W_n} = nI_n$. Das heißt die Fourier-Matrix ist stets invertierbar mit $W_n^{-1} = \frac{1}{n}\overline{W_n}$.

Korollar 1. *Die Diskrete Fouriertransformation f lässt sich auch durch Multiplikation des Datenvektors v der Länge n mit der Fourier-Matrix W_n berechnen.*

$$f = W_n v \Leftrightarrow v = \frac{1}{n}\overline{W_n} f$$

*v in obiger Äquivalenz wird auch **Inverse Diskrete Fouriertransformation** genannt.*

Aus der Invertierbarkeit der Fourier-Matrix folgt, dass die Diskrete Fouriertransformation stets invertierbar ist. Dies kommt uns in der Implementierung zugute, da wir somit ohne Einschränkungen die Signale, die wir zur Verarbeitung in ein Fourierspektrum transformiert haben, stets wieder zu einem Signal zurücktransformieren können.

Es gibt auch einen effizienteren Weg als die direkte Matrixmultiplikation, die Diskrete Fouriertransformation zu berechnen, und zwar durch die Schnelle Fouriertransformation. Für die Schnelle Fouriertransformation braucht man einen Abtastvektor der Länge $n = 2^N$, das heißt die Länge muss einer Zweierpotenz entsprechen. Die Idee des Verfahrens der schnellen Fouriertransformation ist, die einzelnen Berechnungen der Matrix-Vektor-Multiplikation $f = W_n v$ in einer speziellen Reihenfolge auszuführen, sodass jeweils auf schon berechnete Zwischenergebnisse zurückgegriffen werden kann. Hilfreich sind dabei die Eigenschaften der n-ten Einheitswurzel $w_n = exp(\frac{2\pi i}{n})$, dass $w_n^{n/2} = -1$ und dass $w_n^2 = w_{n/2}$.

Korollar 2. ***Schnelle Fouriertransformation***
Für die Schnelle Fouriertransformation betrachten wir die Einträge des Ergebnisvektors $f = (f_1, f_2, f_3, ...)^T$ der Länge n seperat. Sei $N = \frac{n}{2}$
Für die Einträge mit geradem Index gilt: $f_{2j} = \sum_{k=0}^{N-1}(w_N)^{jk}(f_k + f_{k+N})$
Für die mit ungeradem Index: $f_{2j+1} = \sum_{k=0}^{N-1}(w_N)^{jk}(w_n)^k(f_k - f_{k+N})$

Eine Herleitung des Korollar 2 befindet sich in [AHK$^+$12, S.1059].

Satz 3. *Zur Bestimmung der Fourierkoeffizienten aus einem Abtastvektor der Länge $N = 2^p$ benötigt man bei bei der Diskreten Fouriertransformation (DFT) $2N^2 - N$ Rechenoperationen. Bei der Anwendung der Schnellen Fouriertransformation (FFT) liegt der Aufwand bei $\frac{3}{2}Nlog_2N$ Operationen.*

In der Informatik gilt, die DFT hat eine Komplexität von $O(N^2)$, da N^2 der dominierende Term ist. Im Gegensatz dazu funktioniert die FFT zwar nur, wenn die Anzahl der Abtastpunkte einer Zweierpotenz entspricht, braucht aber weitaus weniger Operationen, was bei der Implementierung der Funktion zu einer schnelleren Laufzeit führt. Die FFT hat eine Komplexität von $O(NlogN)$.

Beweis Satz 3
Bei der DFT (16) wird n mal multipliziert und n-1 mal addiert. Dies muss für n Einträge des Datenvektors gemacht werden. Das heißt, man führt insgesamt n^2 Multiplikationen und $n \cdot (n-1) = n^2 - n$ Additionen durch, was zu einer Gesamtzahl von $2n^2 - n$ Operationen führt. Bei der einer Anzahl von $N = 2^p$ Abtastpunkten muss man bei der DFT p Halbierungsschritte durchführen. Nach jeder der p Halbierungen wird 2^p mal subtrahiert bzw. addiert und 2^{p-1} mal multipliziert. Es ergibt sich also eine Gesamtzahl an Operationen von: $p \cdot (2^p + 2^{p-1}) = p2^p + p2^{p-1} = 2p2^{p-1} + p2^{p-1} = 3p2^{p-1}$.
Wegen $N = 2^p$ gilt $log_2N = log_2(2^p) = p$ und die Anzahl der Operationen bezüglich N ergibt sich durch: $3p2^{p-1} = 3p\frac{2^p}{2} = 3log_2N\frac{N}{2} = \frac{3}{2}Nlog_2N$.

Beispiel Fourrierreihenentwicklung

Die Sägezahnfunktion

$$f(x) = \begin{cases} 0, & \text{falls } t \in \{0, 2\pi\}, \\ \frac{1}{2}(\pi - t), & \text{falls } 0 < t < \pi \end{cases}$$

ist ungerade, also gilt : $a_k = 0$ und $b_k = \frac{2}{\pi} \int_0^\pi f(x) sin(kt) dt = \frac{2}{\pi} \int_0^\pi \frac{\pi - t}{2} sin(kt) dt = \frac{1}{k}$
Das reelle Fourierpolynom ist somit gegeben durch
$p_n(x) = a_0 + \sum_{k=1}^n (a_k cos(kx) + b_k sin(kx)) = \sum_{k=1}^n \frac{1}{k} sin(kx)$
Damit ist $p_\infty = \frac{sin(x)}{x} + \frac{sin(2x)}{2} + \frac{sin(3x)}{3} + ...$
Die Amplituden der einzelnen Sinusfunktionen sind somit 1, 1/2, 1/3, ... und dadurch ergibt sich für den Klang, der durch die Sägezahnfunktion dargestellt ist, folgendes Frequenzspektrum:

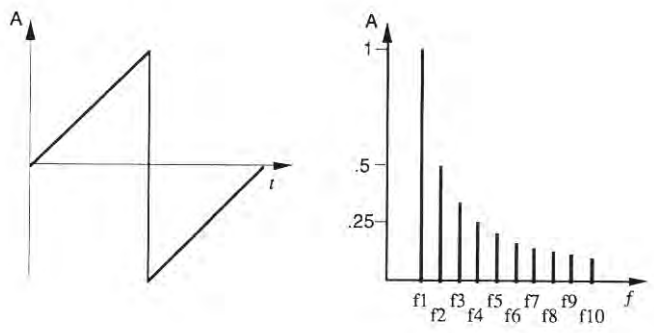

Abbildung 4: links Oszillogramm, rechts Frequenzspektrum der Sägezahnfunktion

Hier sind die Frequenzen $f_1 = \frac{1}{2\pi}, f_2 = \frac{1}{\pi}, f_3 = \frac{1}{\frac{2}{3}\pi} = \frac{3}{2\pi}$ etc.

Beispiel DFT und FFT

Wir betrachten den Datenvekor $v = (3, -2, 0, 1)^T$. Er hat die Länge 4. Es gilt $w_4 = exp(\frac{2\pi i}{4}) = i$, daraus folgt für die DFT anhand der Fouriermatrixmultiplikation:

$$f = W_4 v = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & 1 & -i \end{pmatrix} \cdot \begin{pmatrix} 3 \\ -2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 - 3i \\ 4 \\ 3 + 3i \end{pmatrix}$$

Für die FFT berechnen wir: $\begin{pmatrix} 3 + 0 \\ -2 + 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \end{pmatrix}$ und $\begin{pmatrix} w_4^0(3 - 0) \\ w_4^1(-2 - 1) \end{pmatrix} = \begin{pmatrix} 3 \\ -3i \end{pmatrix}$
$w_2 = exp(\frac{2\pi i}{2}) = -1$, daher gilt:
Für die Koeffizienten f_0 und f_2:

$(3 + (-1)) = (2)$,
$w_2^0(3 - (-1)) = (4)$
Für die Koeffizienten f_1 und f_3:
$(3 + (-3i)) = (3 - 3i)$
$w_2^0(3 - (-3i) = (3 + 3i)$
Und wir erhalten wieder den gleichen Transformationsvektor wie in (22).

[AHK+12, S.1039-1060, S.1171] [Lan]

3.3 Fensterfunktionen

Möchte man aus einem Signal einzelne Zeitabschnitte herausnehmen und untersuchen, muss man diese mithilfe einer Fensterfunktion aus der Signalfunktion extrahieren. Die Abtastdauer des Signals gibt ein Rechteckfenster vor, mit dem ein Ausschnitt des Signals untersucht wird. Sei N die Größe des Fensters. Die Fensterfunktion für das Rechteckfenster lautet:

$$\forall\ 0 \leq n < N - 1 : w(n) = 1 \tag{6}$$

Die zeitliche Begrenzung des Signals durch die Aufnahmedauer ist mathematisch eine Multiplikation mit der Rechteckfunktion (6). Stellt man diese Funktion $f(t) \cdot w(t)$ durch ein Fourierspektrum dar, ergibt sich ein etwas anders Bild als das von $f(t)$. Abbildung 5 visualisiert die Auswirkung der Fensterung auf das Fourierspektrum für ein kontinuierliches Signal.

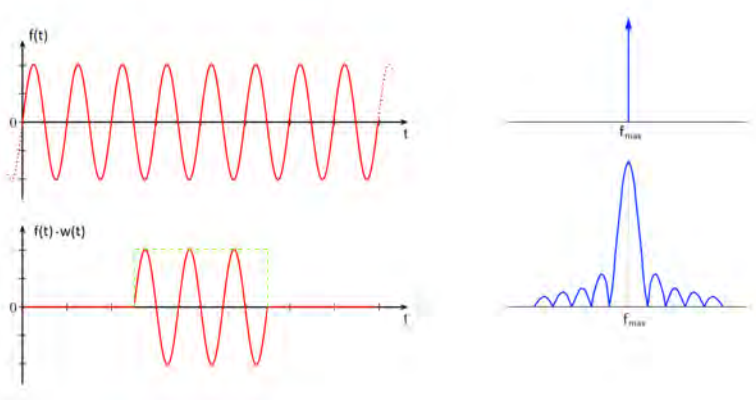

Abbildung 5: Oben Fourierspektrum einer unbeschränkten Funktion, unten Fourierspektrum einer zeitlich beschränkten Funktion

Bei Betrachtung der Abbildung 5 stellt sich die Frage, ob nun nicht auch „falsche“ Frequenzen im Fourierspektrum abgebildet werden. Im kontinuierlichen Fall wird die Frage bejaht. Im diskreten Fall ist dies eine Frage der Abtastung des Fourierspektrums, also der Wahl der Interpolationspunkte bei der DFT. In Abbildung 5

wird deutlich, dass die Nullstellen des Fourierspektrums der gefensterten Sinuskurve den gleichen Abstand zueinander haben, außer an der Stelle f_{max}. Idealerweise findet dann die Abtastung des Fourierspektrums genau bei den Nullstellen und bei f_{max} statt. Solange die Frequenz F des Signals kein ganzzahliges Vielfaches der Abtastfrequenz F_A ist, können sogenannte Leckeffekte bei der DFT auftreten. Ein Leckeffekt entsteht dann, wenn die abzutastende Kurve nicht mehr in ihrem Maximum abgetastet wird und dadurch die Amplitude des Signals beziehungsweise der Signalkomponenten fehlerhaft abgespeichert wird, wie in folgender Abbbildung veranschaulicht:

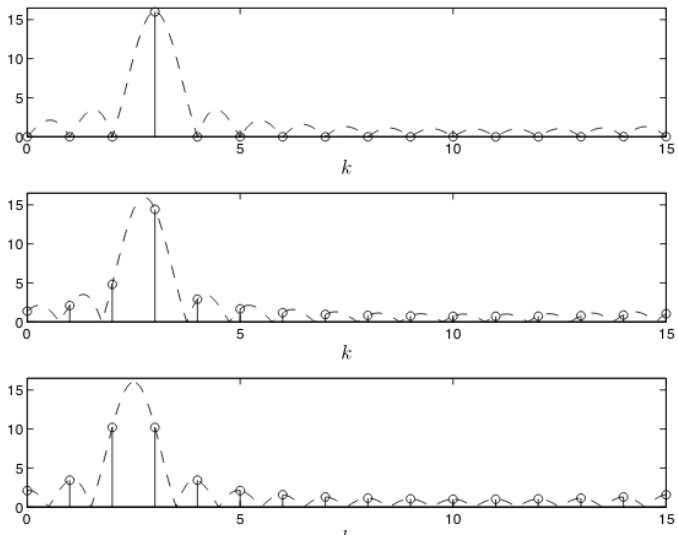

Abbildung 6: Der Leck-Effekt

Da in der Praxis die Abtastfrequenz sehr oft kein ganzzahliges Vielfaches der Frequenz des Signals ist und ein Fehler somit kaum vermeidbar, muss man sich überlegen, wie man den entstanden Fehler beheben kann. Der Fehler, der durch den Leckeffekt entsteht, kann reduziert werden, indem man als Fensterfunktion nicht das Rechteckfenster verwendet, sondern eine andere Fensterfunktion. Diese muss den Anforderungen genügen, dass das Signal im Beobachtungszeitraum bei 0 beginnt und endet, denn dadurch werden die Nebenfrequenzen, also die Frequenzen die eigentlich nicht im Fourierspektrum auftauchen sollten, gedämpft. Zwei Beispiele solcher Fensterfunktionen sind

- Das Hamming-Fenster: $w(n) = 0,54 - 0,46 \cdot cos(\frac{2\pi n}{N-1}) \quad$ für $0 \leq n < N-1$
- Das Gauß-Fenster: $w(n) = e^{-\frac{1}{2}\left(\frac{n-(M-1)/2}{\sigma(M-1)/2}\right)^2} \quad$ mit $\sigma \leq 0,5$ für $0 \leq n < N-1$

In der nachfolgenden Abbildung zeigt sich der Einfluss der Fensterfunktion auf die Eindämpfung der Nebenfrequenzen. [Ack91, S.93] [PKJ11, S.266]

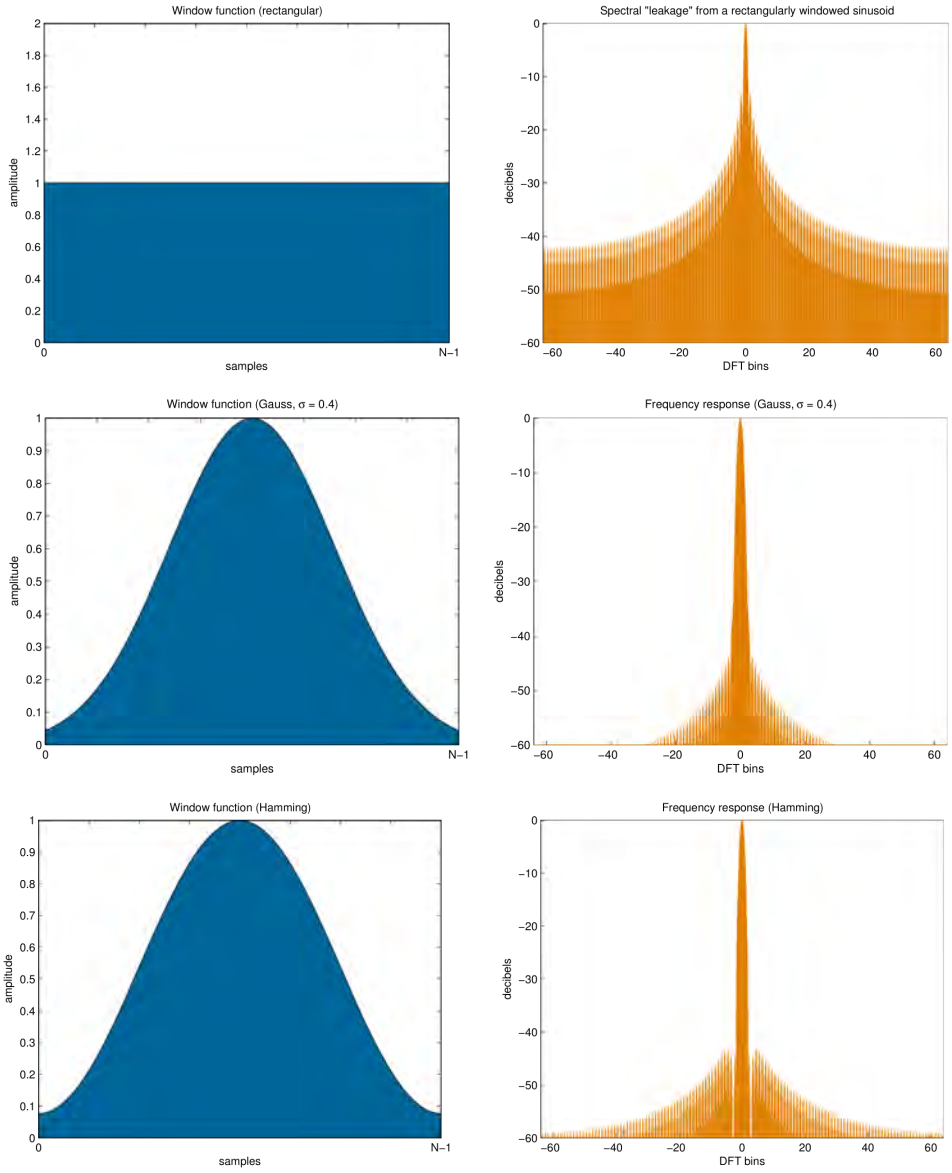

Abbildung 7: Vergleich der Fourierspektren verschiedener Fensterfunktionen

3.4 Das Spektrogramm

Zur Manipulation und Mustererkennung an Audiosignalen ist es hilfreich, am Spektrogramm des Signals zu arbeiten. In Kapitel 2.3 wurde bereits erwähnt, dass ein Fourierspektrum eine Amplituden-Frequenz-Darstellung ist. Ähnlich dazu ist ein Spektrogramm die dreidimensionale Darstellung des Singals, und zwar in den Dimensionen Amplitude, Frequenz und Zeit.

Das dreidimensionale Spektrogramm kann in ein zweidimensionales Bild umgeformt werden, indem die Achsen Zeit und Frequenz an eine Matrix angebracht werden und jeder Eintrag der Matrix die Amplitude beschreibt. Die Zahlen werden daraufhin grafisch durch eine Farbabstufung sichtbar gemacht, z.B. werden Punkte dunkler eingefärbt je höher die Amplitude ist. Da die Audiodatei jetzt als Bild vorliegt, bietet

es sich an, Verfahren auf dem Spektrogramm zu testen, die auch in der Bildverarbeitung verwendet werden, z.B. die nonnegative Matrixfaktorisierung.
Da das Spektrogramm quasi ein Fourierspektrum mit zusätzlicher Zeitachse ist, muss das Signal in Zeitabschnitte eingeteilt werden. Diese Zeitabschnitte dürfen sich überlappen und werden mithilfe einer Fensterfunktion aus dem Signal entnommen. Auf jedem dieser Zeitabschnitte wird eine Fouriertransformation durchgeführt und man erhält einen Vektor pro Zeitabschnitt. Daraus wird eine Matrix gebildet mit einer Zeile je Zeitabschnitt, in der in jeder Spalte die Frequenzanteile, also die Koeffizienten der Summanden der Fourierreihe, eingetragen werden. [Vir07]
In den Abbildungen 8, 9 und 10 sind drei verschiedene Visualisierungformen der Audiodatei *SOUNDSEPP\input\piano_kick_monoton* (befindet sich auf beiliegendem Datenträger) zu sehen.

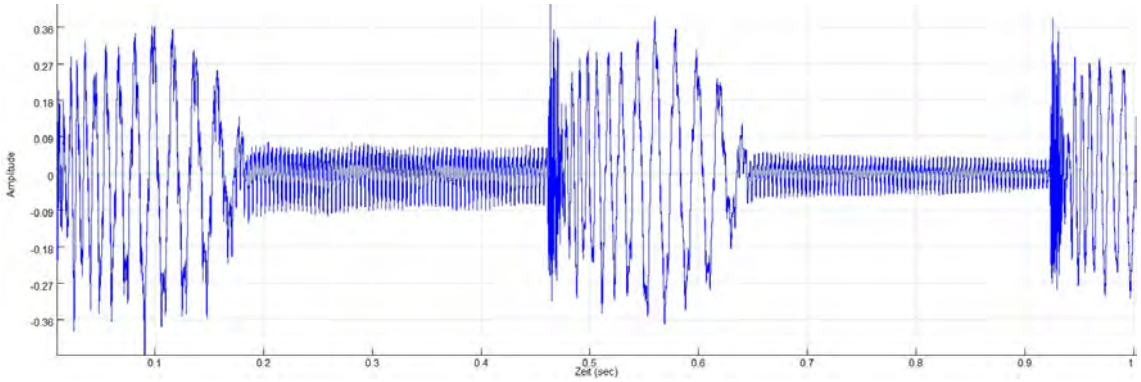

Abbildung 8: Oszillogramm

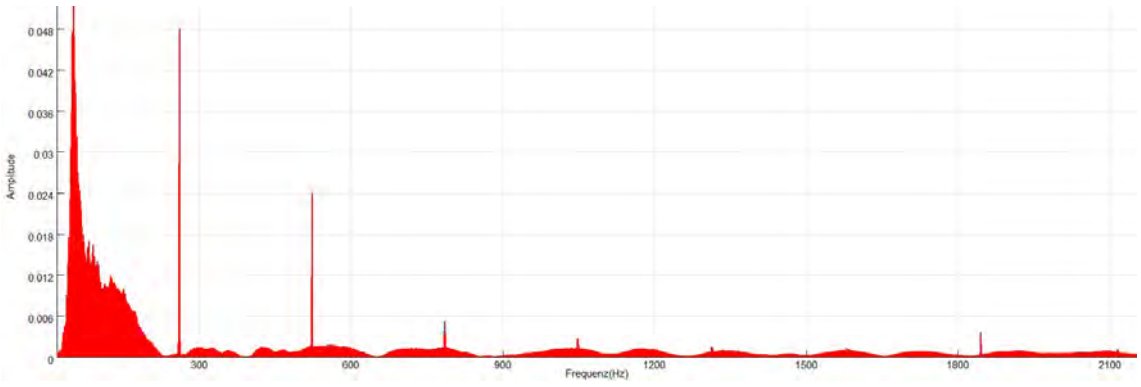

Abbildung 9: Fourierspektrum

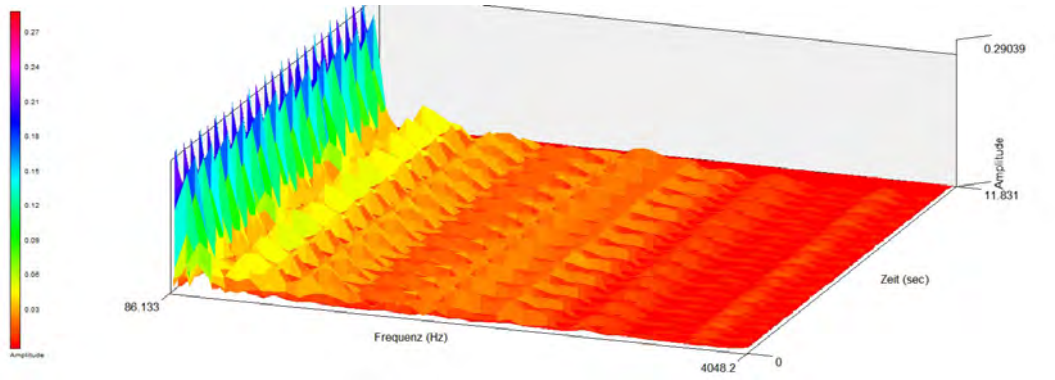

Abbildung 10: dreidmensionales Spektrogramm

In der folgenden zweidimensionalen Version des Spektrogramms (Abbildung 11) kann man mit geübtem Auge ungefähr das Aufkommen und die Art der Quellen ablesen. Die gelbe Fläche symbolisiert die harmonischen Schwingungen des Pianos, die sich der Abbildung 11 zufolge bei ungefähr 300 Hz hauptsächlich aufhalten. Der Kammerton a hat eine Frequenz von 440 Hertz, das Mittlere c befindet sich bei circa 260 Hertz. Das heißt, man kann ohne die Audiodatei zu hören feststellen, dass sehr wahrscheinlich ein harmonisches Instrument dabei ist, das dreimal hintereinander den gleichen Klang spielt, der irgendwo zwischen dem mittleren c und a liegt. Die ständig sich wiederholenden Spitzen ohne eindeutige Frequenz entstehen durch die Kick-Drum.

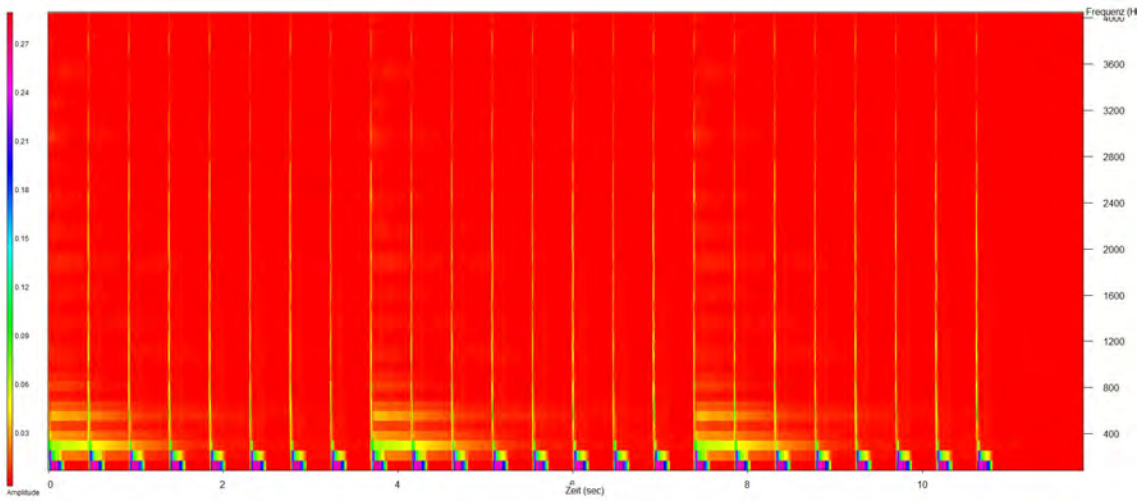

Abbildung 11: zweidimensionales Spektrogramm

Berechnung eines Spektrogrammes

Für alle nachfolgenden Verwendungen in dieser Arbeit wird für das Bilden der Spektrogramm-Matrix folgendes festgelegt:

1. Die Abtastrate F_A des Signals beträgt 44100 Hz.
2. Das Signal wird in Zeitabschnitte der Länge 40 ms mit einer Überlappung von 50% zerteilt. Das heißt, in jedem Zeitabschnitt gibt es $\frac{40}{1000} \div \frac{1}{44100} = 1764$ Abtastpunkte.
3. Jeder dieser Zeitabschnitte wird durch das Hamming-Fenster aus dem Signal entnommen.
4. Sei T Anzahl der Zeitfenster, N = 1764 Länge der Zeitfenster und $F_{max} = \frac{F_A}{2}$ die maximal auftretende Frequenz gemäß dem Nquist-Shannon-Theorem. Weiterhin sei $t \in \{0, ..., T-1\}$, $n \in \{0, ..., N-1\}$ und $f \in \{0, ..., F_{max} - 1\}$. Die FFT für ein diskretes Signal x wird dann folgendermaßen berechnet:

$$X(f,t) = \sum_{n=0}^{N-1} (0,54 - 0,46cos\frac{2\pi n}{N}) x(n + t \cdot \frac{N}{2}) exp(\frac{-j2\pi fn}{N}) \tag{7}$$

Die Einträge $X(f,t)$ des Spektrogramms geben die Amplitude der Frequenz $\tilde{f}$ zum Zeitpunkt $\tilde{t}$ an. Genauer gesagt ist t der Index des Fensters, dass zur Zeit $\tilde{t} = t \cdot \frac{N}{2F_A}$ in Sekunden startet und f ist die Frequenz $\tilde{f} = \frac{f}{N} \cdot F_A = \frac{f}{1764} \cdot 44100 = 25f$ in Hertz. Die Amplituden sind nach der Transformation (7) komplex. Da $\mathbb{C}$ keine geordnete Menge ist, kann man das Spektrogramm aber nicht wie in Abbildung 11 darstellen, weil man keinen linearen Zusammenhang zwischen Farbe und Wert konstruieren kann. Betrachtet man die komplexen Einträge des Spektrogramms in Polardarstellung $z = |z|(cos(\phi) + i \cdot sin(\phi))$, so gibt das Argument ϕ die Phase der Schwingung und der Betrag $|z|$ die maximale Auslenkung der Schwingung an. Da die Phase für das Gehör unwichtig ist, kann man zum Veranschaulichen das Argument der Zahl einfach durch Berechnen des Betrags von z verwerfen und erhält eine reelle positive Zahl. Die Argumente der Einträge sollten in einem extra Phasen-Spektrogramm arg(X) abgespeichert werden.
$|X|$ wird in der Fachliteratur „Magnitude Spektrogramm“ genannt, dieses ist letztendlich graphisch visualisierbar. Aus dem Magnitude-Spektrogramm kann man das Signal wieder durch Hinzunahme des Phasen-Spektrogramms konstruieren, denn

$$X = |X| \cdot (cos(arg(X)) + i \cdot sin(arg(X))). \tag{8}$$

$|X|^2$ nennt man „Power-Spektrogramm“. Die englischen Begriffe „magnitude“ und „power“ lassen sich beide mit „Stärke“ ins Deutsche übersetzen, weshalb im Folgenden von einer Übersetzung der Fachbegriffe abgesehen wird. Bei Power-Spektrogram-

men werden durch die Quadrierung zusätzlich größere Einträge stärker gewichtet, was für gewisse Verwendungszwecke vorteilig sein kann.
Zwei Signale x_1 und x_2 summieren sich linear, denn das Mischsignal x_{12} für x_1 und x_2 ergibt sich durch $x_{12} = x_1 + x_2$. Für die Spektrogramme X_1 von x_1 und X_2 von x_2 gilt ebenso, dass das Spektrogramm X_{12} des Mischsignals durch $X_{12} = X_1 + X_2$ gebildet werden kann. Dies gilt allerdings nicht immer für die Magnitude- und Power-Spektrogramme, Grund dafür sind die Phasen der Signale.
Es ist ratsam, für den Zweck der Signaltrennung mithilfe von unüberwachten Lernalgorithmen das Magnitude-Spektrogramm zu verwenden, da bei der Magnitude die bezüglich des Betrages kleineren Einträge nicht so sehr vernachlässigbar gemacht werden wie beim Power-Spektrogramm. Für numerische Anwendungen und somit auch für Lern-algorithmen gilt pauschal, dass ein Eintrag umso vernachlässigbarer ist umso kleiner er ist. Jedoch ist für die akustische Wahrnehmung von Signalen auch ein kleiner, also „leiser“ Eintrag des Spektrogramms sehr wohl bedeutsam. [MEKR11] [KD06, S.273f]

Mathematische Modellierung der Signalkomponeten mithilfe der Spektrogramms

Sei $X = [x_1, ..., x_T]$ ein Magnituden-Spektrogramm eines Audiosignals. Sei

$$X = BG, \quad B \in \mathbb{R}_+^{F \times K}, \quad G \in \mathbb{R}_+^{K \times T} \tag{9}$$

Eine Methode, mit der man eine derartige Faktorisierung von X annäherungsweise finden kann, sodass $X \approx BG$, ist die Nonnegative Matrixfaktorisierung, die in Kapitel 3.6 genauer erläutert wird.
g_k sind nun die Spaltenvektoren von G und b_k die Zeilenvektoren von B mit $k \in \{1, .., K\}$. b_k kann als Spektral-Basis gesehen werden und g_k als die zugehörige zeitliche Lautstärkeveränderung. Deshalb wird B Basis-Matrix und G Zunahme-Matrix genannt. Das Magnituden- Spektrogramm einer Quellkomponente ist dann:

$$X_k = g_k b_k$$

Um das Magnituden-Spektrogramm des Quellsignals X_s zu erhalten, muss man eventuell Quellkomponenten addieren. Es gibt keine Methode, mit der man die Anzahl K mithilfe des Spektrogramms eindeutig festlegen kann. Daher muss man k manuell wählen und eventuell manuell festlegen, welche X_k zusammengehören und addiert werden müssen.
Bei der Rekonstruktion der Quellsignale X_s ergibt sich das Problem, dass man zur Rekonstruktion mithilfe der inversen Fouriertransformation die originalen Spektrogramme braucht, und nicht bloß die Magnituden-Spektrogramme. Der einfachste

Weg hierbei ist, sich des Phasen-Spektrogramms der Audiodatei zu bedienen, die man zerlegt und Formel (8) anwendet. In den meisten Fällen liefert das schon gute Ergebnisse. Falls nicht, gibt es auch einen Algorithmus von Griffin und Lim, mit dem man die benötigten Phasen generieren kann. [Vir07]

3.5 Optimierung

Nun steht einem das Magnituden-Spektrogramm X zur Verfügung, und man möchte eine Zerlegung in $X = BG$ wie in (9) finden. Dazu wird eine Funktion f benötigt, die die Differenz zwischen X und BG für verschiedene BG auswertet, also $f(X, BG) \in \mathbb{R}$. Diese heißt Kostenfunktion. Ist die Kostenfunktion für X an einem BG an ihrem globalen Minimum, so ist dieses BG die beste Approximation mit $X \approx BG$. [LS01] Allgemein spricht man bei solchen Vorgehen von Optimierungsaufgaben.

Definition 11. *Eine* ***Optimierungsaufgabe*** *liegt vor, wen zu einer Abbildung $f : D \to \mathbb{R}$ Elemente $y \in D \subseteq \mathbb{R}^n$ gesucht sind, die f minimieren, das heißt*

$$f(y) \leq f(x) \quad \text{für alle } x \in D$$

Um diese Schreibweise abzukürzen, notiert man Optimierungsprobleme durch

$$min_{x \in D} f(x)$$

Manchmal ist es nötig, zur Lösung von Optimierungsaufgaben numerische Annäherungsverfahren anzuwenden. Das heißt man wählt einen zufälligen Startwert $x_0 \in D$ und geht von diesem eine gewisse Schrittweite t in Richtung eines Abstiegs p < 0 der Funktion und iteriert dies solange, bis man an einem lokalen Minimum angekommen ist. Der erste Schritt muss also erfüllen:

$$f(x_0 + tp) < f(x_0)$$

Ein Kriterium für eine Abstiegsrichtung p ist $\nabla f(x) \cdot p < 0$, denn

$$\nabla f(x) \cdot p = \lim_{t \to 0} \frac{f(x + tp) - f(x)}{t} < 0.$$

Der nun folgende Algorithmus zur Suche des lokalen Minimums nennt sich Gradientenverfahren. Ausgehend von einem Startwert x_0 bestimmt man für $k = 0, 1, 2, ...$ den nächsten Iterationsschritt x_{k+1}, indem man zu x_k

1. testet, ob $0 \leq \nabla f(x_k) \cdot p$ für alle Richtungen $p \in D$ gilt. Ist dies so, sind wir an einem lokalen Minimum gelandet und brechen die Iteration sofort ab.

2. eine Abstiegsrichtung p_k wählt mit der Eigenschaft $\nabla f(x_k) \cdot p < 0$

3. eine Schrittweite t_k festlegt und $x_{k+1} = x_k + t_k \cdot p$ berechnet.

Für die Abstiegsrichtung ist es naheliegend, einfach den negativen Gradienten zu wählen, also $p_k = -\nabla f(x_k)$, denn dieser gibt nicht nur irgendeine Abstiegsrichtung vor, sondern er zeigt den steilsten Abstieg. Es gilt nämlich nur für $||d|| = 1$ stets:

$$d \cdot \nabla f(x) \leq ||d|| \cdot ||\nabla f(x)|| = ||\nabla f(x)|| \quad \forall x \in D$$

[AHK+12, S.1196, S.1215ff]

3.6 Nonnegative Matrixfaktorisierung (NMF)

Eine Möglichkeit, die Signale so zu zerlegen, dass durch die Zerlegung nichts oder kaum etwas verloren geht, ist durch Matrixfaktorisierung. Gegeben ist die Matrix des Spektrogramms. Durch Matrixfaktorisierung erhält man mehrere Matrizen, die miteinander multipliziert die Matrix ergeben. Eine Matrixmultipilikation kann aber auch als Summe aufgefasst werden, indem man nicht wie üblich Zeilenvektoren mit Spaltenvektoren multipliziert und so einen einzelnen Eintrag erhält, sondern indem man Spaltenvektoren mit Zeilenvektoren multipiliziert und die dadurch entstehenden Teilmatrizen miteinander addiert. Diese Teilmatrizen könnten dann bei der Signaltrennung jeweils das Spektrogramm einer einzelnen Komponente sein.

3.6.1 Theorie der NMF

NMF funktioniert nur auf nichtnegativen Matrizen, das heißt für die Matrix X muss $X(f,t) > 0$ für alle $(f,t) \in F \times T$ gelten. Ziel der NMF ist es, eine Faktorisierung für $X \in \mathbb{R}_+^{F \times T}$ zu finden, sodass

$$X \approx BG, \quad B \in \mathbb{R}_+^{F \times K}, \quad G \in \mathbb{R}_+^{K \times T} \tag{10}$$

X, B und G sind dabei eintragsweise nicht negativ. K ist beliebig wählbar mit $K < min(F,T)$. K gibt die Anzahl der Komponenten an, in die die Datei getrennt werden soll. Es gibt keinen Algorithmus, mit dem B und G in einer Faktorisierung gemäß (10) exakt berechnet werden können, sondern B und G müssen in einem Optimierungsverfahren approximiert werden. Die Güte der Approximation wird durch eine Kostenfunktion gemessen. Beispiele für passende Kostenfunktionen zur NMF sind:

- Der euklidische Abstand $E(X||BG) = \sum_{(f,t)\in F\times T}(X(f,t) - BG(f,t))$ (gibt nur den Rekonstruktionsfehler an)
- Die Kullback-Leibler-Divergenz $D(X||BG) = \sum_{(f,t)\in F\times T} X(f,t) log \frac{X(f,t)}{BG(f,t)}$ (gibt nur den Rekonstruktionsfehler an)

- Eine Mischung aus Messwerten für Rekonstruktionsfehler, zeitlicher Kontinuität und Dünnbesetztheit, damit zeitlich kontinuierliche und dünnbesetzte Matrizen favorisiert werden können. (Genauer darauf eingegangen wird in Kapitel 4)

Für die Berechnung von B und G gibt es verschiedene unüberwachte Lernalgorithmen. Sie sind Implementierungen einer Optimierungsaufgabe, welche fordert, dass die gewählte Kostenfunktion konvergiert, das heißt bei jeder Iteration der Schätzung der Matrizen B und G kleiner wird. Also in der Notation von Optimierungsaufgaben:

$$min_{BG \in \mathbb{R}_+^{F \times T}} E(X||BG) \quad bzw. \quad min_{BG \in \mathbb{R}_+^{F \times T}} D(X||BG)$$

Folgender Algorithmus kann verwendet werden, wenn für die Kostenfunktion die Kullback-Leibler-Divergenz oder der euklidischen Abstand gewählt wird:

1. Wähle K beliebig und initialisiere die Einträge B und G. Die Güte der Approximation hängt auch von der Wahl der Initialisierung ab. Welche Initialisierungsschemata man verwenden kann, wird in Kapitel 3.6.3 genauer erläutert.

2. Aktualisiere B so, dass die Kostenfunktion konvergiert. Zum Beispiel:

   Für die Kullback-Leibler Divergenz: $B \leftarrow B. \times \frac{\frac{X}{BG}G^T}{\mathbf{1}G^T}$

   Für den euklidischen Abstand: $B \leftarrow B. \times \frac{XG^T}{BGG^T}$

3. Aktualisiere G so, dass die Kostenfunktion konvergiert. Zum Beispiel:

   Für die Kullback-Leibler Divergenz: $G \leftarrow G. \times \frac{B^T\frac{X}{BG}}{B^T\mathbf{1}}$

   Für den euklidischen Abstand: $G \leftarrow G. \times \frac{B^TX}{B^TBG}$

4. Wiederhole Schritt 2 und 3 solange bis die Kostenfunktion konvergiert.

Hierbei bedeutet $A \times B$ und A/B die elementweise Muliplikation beziehungsweise Division der Matrizen A und B. Die Aktualisierung ist wohldefiniert, denn es wird nicht durch Null geteilt, weil B und G eintragsweise nicht negativ sind. $\mathbf{1}$ ist eine nur mit Einsen gefüllte Matrix der Größe $F \times T$. Es können auch andere Aktualisierungsregeln formuliert werden, es muss bloß sichergestellt werden, dass die Kostenfunktion durch die Aktualisierung konvergiert. Einen Beweis für die Konvergenz der Kostenfunktion bei obigen Lernalgorithmen findet man in [LS01].
Hinter obigem Algorithmus für die Kullback-Leibler Divergenz und die euklidische Distanz steckt ein Gradientenverfahren. [KD06, S.282]

Herleitung der Aktualisierungsregeln für die euklidische Distanz

Sei nun in der Formel (3.5) f durch die euklidische Distanz gegeben. x_0 sei die anfangs beliebig initialisierte Matrix G. Als Abstiegsrichtung wählt man $(B^TX - B^TBG)$, denn für $X = BG$ wäre $B^TX = B^TBG$ und somit wäre gewährt, dass nicht mehr weiter abgestiegen wird, wenn die exakte Lösung gefunden wurde. Wählt man nun für die Schrittweite $t = \frac{G}{B^TBG}$, wobei auch hier die eintragsweise Division gemeint ist, so ergibt sich $G \leftarrow G + \frac{G}{B^TBG}(B^TX - B^TBG) = G + \frac{G.\times B^TX}{B^TBG} - \frac{G.\times B^TBG}{B^TBG} = G + \frac{G.\times B^TX}{B^TBG} - G = G. \times \frac{B^TX}{B^TBG}$. Analog wird der Iterationsschritt von B berechnet. Verfährt man nun wie beim Gradientenverfahren, ergibt sich genau der Algorithmus für die euklidische Distanz aus diesem Kapitel. [LS01]

3.6.2 Beispielrechnung

Sei $X = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$, $G = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

Die Aktualisierungsregel für den euklidischen Abstand liefert:

1. Iteration:

$B \leftarrow B. \times \frac{XG^T}{BGG^T} = \begin{pmatrix} 1 & 1 \\ 2,5 & 2,5 \\ 4 & 4 \end{pmatrix}$ und $G \leftarrow G. \times \frac{B^TX}{B^TBG} \approx \begin{pmatrix} 0,84 & 1 & 1,16 \\ 0,84 & 1 & 1,16 \end{pmatrix}$

$B \cdot G \approx \begin{pmatrix} 1,68 & 2 & 2,23 \\ 4,19 & 5 & 5,81 \\ 6,71 & 8 & 9,29 \end{pmatrix}$

Somit ist der Rekonstruktionsfehler nach der ersten Iteration $E(X||BG) \approx 2,23$

2. Iteration und 100.Iteration:

$B \leftarrow B. \times \frac{XG^T}{BGG^T} \approx \begin{pmatrix} 1,04 & 1,04 \\ 2,51 & 2,51 \\ 3,98 & 3,98 \end{pmatrix}$ und $G \leftarrow G. \times \frac{B^TX}{B^TBG} \approx \begin{pmatrix} 0,84 & 1 & 1,16 \\ 0,84 & 1 & 1,16 \end{pmatrix}$

$B \cdot G \approx \begin{pmatrix} 1,74 & 2,07 & 2,41 \\ 4,21 & 5,02 & 5,83 \\ 6,68 & 7,97 & 9,26 \end{pmatrix}$

Somit ist der Rekonstruktionsfehler nach der zweiten und hundertsten Iteration $E(X||BG) \approx 2,4$. Tatsächlich liefern diese Werte zufällig sogar ein Beispiel dafür, dass sich der Rekonstruktionsfehler zwischenzeitlich auch mal verschlechtern kann. Konvergieren tut er trotzdem, denn er pendelt sich nach egal wie viel Iterationen bei 2,4 ein.

3.6.3 Initialisierung der Startmatrizen

Es gibt eine Fülle von Ansätzen für die Initalisierung der Startmatrizen B und G. Sich darüber Gedanken zu machen wie man die Startmatrizen initialisiert, ist deshalb wichtig, weil die Wahl der Startmatrizen die Qualität des Ergebnisses maßgeblich beeinflussen kann und sich auf die Laufzeit des Algorithmus auswirkt. Initialisiert man die Basismatrix B mit vordefinierten Basisvektoren, die spezifische Instrumente repräsentieren, verkürzt sich die Laufzeit enorm, so wie in [DM16] aufgezeigt. Von einer Initialisierung aller Einträge der Startmatrizen mit dem gleichen Wert sollte allerdings abgesehen werden, betrachte dazu die Beispielrechnung in Kapitel 3.5.2. Hier wird deutlich, dass die einzelnen Basisvektoren stets gleich bleiben, was bei der Signaltrennung mithilfe der NMF dazu führen würde, dass wir das Signal in identische Teilsignale zerlegen würden.
Zufällig initialisierte Startmatrizen liefern akzeptable Ergebnisse, wobei hier auch die Wahrscheinlichkeitsverteilung der Zufallszahlen eine Rolle spielt. Man kann sich beispielsweise zwischen der diskret gleichverteilten Zufallszahlen oder der Standartnormalverteilung entscheiden.

Satz 4. *diskrete Gleichverteilung*
Sei n die Anzahl der speicherbaren Zahlen zwischen 0 und 1. Diese sind von der Auflösung des Computers bestimmt. Ist die Zufallsvariable X im Intervall [0,1] diskret gleichverteilt, so gilt für k = 0,...,1 für die Wahrscheinlichkeit

$$P(X = k) = \frac{1}{n}$$

Satz 5. *Standartnormalverteilung*
Die Standartnormalverteilung ist eigentlich eine stetige Verteilung. Der Maßstab für die Wahrscheinlichkeit bei stetigen Verteilungsfunktionen ist die sogenannte Wahrscheinlichkeitsdichte, diese ist bei der Standartnormalverteilung folgendermaßen definiert

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}}$$

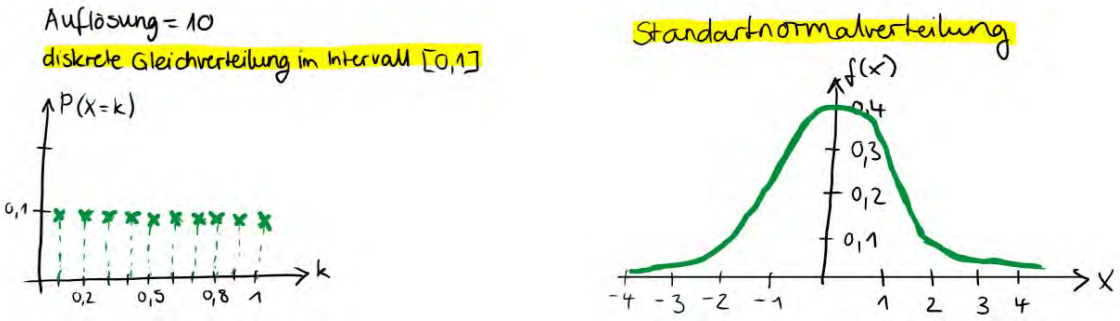

Abbildung 12: Vergleich der Gleichverteilung mit der Standartnormalverteilung

Für die Initialisierung der Matrizeneinträge mithilfe der Standartnormalverteilung greift man nun diskret für jeden einzelnen Eintrag einen Wert aus der Standartnormalverteilung heraus und nimmt davon den absoluten Betrag. Wie in obiger Abbildung 12 ersichtlich, liegt dann der Großteil der zufällig generierten Zahlen dann zwischen 0 und 1 und je kleiner die Zahl ist, desto öfter kommt sie vor. [AHK+12, S.1347]

3.6.4 Singulärwertzerlegung im Gegensatz zur NMF

Die Singulärwertzerlegung kann auf jeder beliebigen – auch komplexen – Matrix durchgeführt werden. Sie ist keine Annäherung einer Faktorisierung, sondern exakte Faktorisierung einer Matrix.

Satz 6. *Singulärwertzerlegung einer Matrix*
Für jede Matrix $A \in \mathbb{C}^{n\times m}$ gibt es orthogonale Matrizen $U \in \mathbb{C}^{n\times n}$ und $V \in \mathbb{C}^{m\times m}$ mit

$$A = U^T DV$$

*Diese Darstellung heißt **Singulärwertzerlegung** von A. Die Quadrate der in der Hauptdiagonale von D auftauchenden Singulärwerte $s_1, ..., s_r$ von A sind die von null verschiedenen Eigenwerte der symmetrischen Matrix $A^T A$. Die Vielfachheit des Singulärwertes gibt an, wie oft der jeweilige Singulärwert aufzuschreiben ist.*

Die Singulärwerte von A und somit die Matrix D sind eindeutig bestimmt – U und V hingegen nicht. Da U und V unitäre Matrizen sind, repräsentieren sie als Abbildung gesehen Bewegungen (Drehungen, Spiegelungen und Verschiebungen). Die Matrix D ist dadurch, dass sie nur auf der Diagonale Einträge hat, eine Skalierung. Mit der Singulärwertzerlegung kann man durch U die Korrelation der Quellvektoren – hier der Quellsignale – mit den Beobachtungsvektoren – hier dem Originalsignal –, durch V die Korrelation der Beobachtungsvektoren mit den Quellvektoren und durch D den Einfluss der einzelnen Vektoren modellieren. Ein sehr großer Nachteil ist aber, die Anzahl der Quellvektoren mit denen der Beobachtungsvektoren übereinstimmen muss und deshalb ungeeignet für Mono-Sound-Dateien ist. Bei Mono-Sound-Dateien gibt es nur einen Beobachtungsvektor. Dieses Problem kann man umgehen, indem man das Spektrogramm der Mono-Sound-Datei in Frequenz-Rahmen einteilt, und somit mehrere Beobachtungsvektoren erhält. Das ist der Schlüssel für die Independent Subspace Analysis (ISA), die auf der Theorie der Singulärwertzerlegung aufbaut. Bei der ISA werden die Signale so getrennt, dass sie maximal statistisch unabhängig (englisch: independent) voneinander sind, und dies ist laut Kapitel 2.7 ein Mustererkennungskriterium in Audiosignalen. Andere Eigenschaften von Audiosignalen wie z.B. Kontinuität und somit Anforderungen an eine Signaltrennung lassen sich aber mit der ISA nicht umsetzen. [AHK+12, S.729] [KD06, S.274]

4 Signaltrennung mithilfe Nonnegativer Matrixfaktorisierung unter Berücksichtigung der zeitlichen Kontinuität und Dünnbesetztheit der Faktoren

Im Folgenden wird der von Tuomas Virtanen [Vir07] vorgeschlagene Ansatz zur Signaltrennung erläutert. Er ist eine Erweiterung der Nonnegativen Matrixfaktorisierung um ein Gradientenverfahren, das sowohl den Rekonstruktionsfehler als auch die zeitliche Kontinutät und Dünnbesetztheit der Signale miteinbezieht.

4.1 Ablauf des Signaltrennungsverfahrens

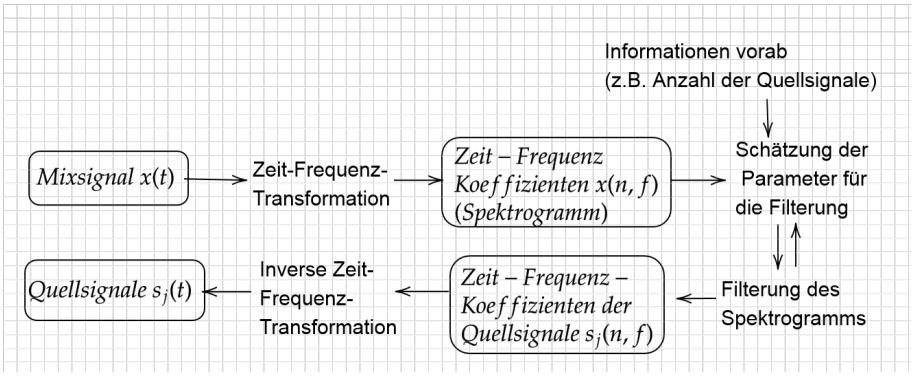

Abbildung 13: Ablaufdiagramm der Signaltrennung

Die meisten Signaltrennungsverfahren verfolgen den Verlauf in Abbildung 13 [VVG18, S.9]. Auch der Ansatz von Toumas Virtanen verwendet diese Vorgehensweise. Als Zeit-Frequenz-Tranformation schlägt er die Diskrete Fouriertransformation (DFT) vor. Das Signal x(t) wird durch die DFT in ein Spektrogramm $X \in \mathbb{C}^{FxT}$ umgewandelt. In der Implementierung ist es aber ratsam, die FFT zu verwenden, da sie dasselbe berechnet und weniger Laufzeit braucht. Da die NMF nur auf nichtnegativen Matrizen funktioniert, verwandelt er das Spektrogramm vor der Filterung noch in ein Magnituden-Spektrogramm, indem er einfach die absoluten Werte der Einträge verwendet. Allgemein werden die Spektrogramme der resultierenden Quellkomponenten durch die Wahl der Filterung bestimmt. Um ein bestmögliches Ergebnis zu erhalten, werden die Werte der Parameter zur Filterung erst geschätzt – also die Startmatrizen initalisiert – und immer wieder aktualisiert, bis man mit den Ergebnissen der Filterung zufrieden ist. Man erhält aus der Filterung die Zeit-Frequenz-Koeffizienten der Quellsignale. Bei Virtanen hat man nach Durchführung seines Filteralgorithmus nur den Absolutbetrag derer, aber wenn man diese wieder in die komplexe Form mithilfe einer Phasenrekonstruktion bringt, können sie durch die IFFT in die Quellsignale s(t) verwandelt werden.

4.2 Die Kostenfunktion

Die Kostenfunktion dieses NMF-Ansatzes zur Erlangung einer Approximation $X \approx BG$ ist von der zeitlichen Kontinutät $c_t(G)$, der Dünnbesetztheit $c_s(G)$ und dem Rekonstruktionsfehler $c_r(B, G)$ abhängig. Sie hat folgende Form:

$$c(B, G) = c_r(B, G) + \alpha c_t(G) + \beta c_s(G) \tag{11}$$

Hierbei ist α und β die Gewichtung der einzelnen Terme. Virtanen hat für die Gewichtung verschiedene Werte durchgetestet und herausgefunden, dass mit $\alpha = 100$ und $\beta = 0$ durchschnittlich die besten Trennungsergebnisse erzeugt werden.

Rekonstruktionsfehler

Für den Rekonstruktionsfehler von X und BG verwendet Virtanen die eine an die Kullback-Leibler-Divergenz angelehnte Formel:

$$c_r(B, G) = \sum_{(f,t) \in F \times T} (X(f,t) log \frac{X(f,t)}{BG(f,t)} - X(f,t) + BG(f,t)) \tag{12}$$

Die Kullback-Leibler-Divergenz ist eigentlich ein Maß für die Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen. Es liegt nahe für Rekonstruktionsfehler auch mit den euklidischen Abständen zu rechnen. Allerdings sollten große Unterschiede zwischen den Einträgen von X und BG stark ins Gewicht fallen, während kleine Unterschiede eher vernachlässigbar sind. Dies kann erreicht werden, indem man das Quadrat des euklidischen Abstands verwendet, was zu folgender Metrik führt: $E(X||BG) = \sum_{k,t}([X]_{k,t} - [BG]_{k,t})^2$. Offensichtlich gilt $E(\gamma X||\gamma BG) = \gamma^2 E(X||BG)$ wohingegen $D(\gamma X||\gamma BG) = \gamma D(X||BG)$. Wir erinnern uns, dass die Einträge der Matrizen die Lautstärke der Frequenz f zum Zeitpunkt t angeben. Somit ist die Divergenz (12) bei *leisen* Signalen das Mittel der Wahl, weil $E(X||BG)$ unsensibler gegenüber kleinen Werten als $D(X||BG)$ ist. Da leise Signale in der Wahrnehmung und auch für die musikalische Gesamtheit bedeutsam sind, ist es nachvollziehbar, dass auch bei Virtanen die Formel (12) die besseren Resultate geliefert hat.

Zeitliche Kontinuität

Folgende Abbildung 14 zeigt ein Spektrogramm in der Mitte, das man in zwei Quellkomponenten trennen will. Man könnte das Spektrogramm in der Mitte sowohl in Spektrogramm A und B zerteilen als auch in Spektrogramm C und D. Verglichen mit dem Spektrogramm in Kapitel (3.4) liegt die Vermutung nahe, dass das Spektrogramm in der Mitte eine Mischung aus einem Percussioninstrument und einem harmonischen Instrument ist und dass die Trennung in Spektrogramm A und B in den meisten Fällen die richtige Trennunng ist. Mithilfe dieser Abbildung 14 kann

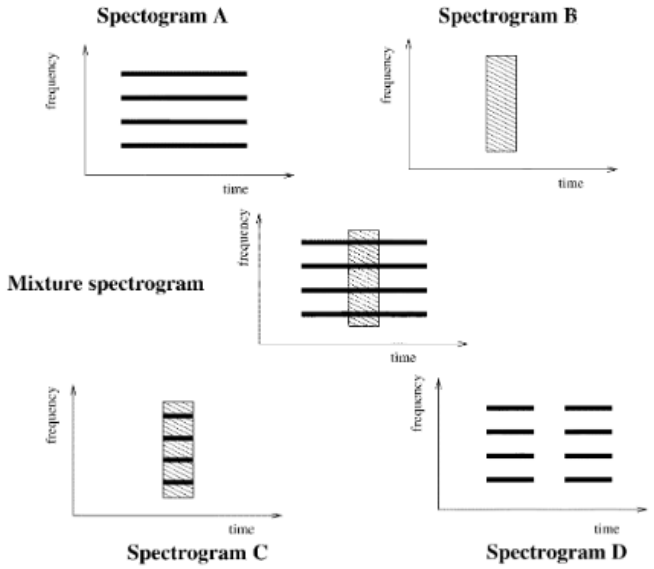

Abbildung 14: Beispiel für die zeitliche Kontinuität von Spektrogrammen

man also sehr deutlich erkennen wie wichtig es ist, in der Kostenfunktion zeitlich kontinuierliche Signale zu favorisieren. Favorisieren heißt, dass die Kostenfunktion, die man beim Optimierungsverfahren minimieren möchte, nach einer Iteration ist umso geringer ausfallen sollte, je zeitlich kontinuierlicher die Signale sind.
Die Basismatrix $B \in \mathbb{C}^{F \times K}$ enthält keine Information über den zeitlichen Ablauf der Komponenten. Deshalb misst man die zeitliche Kontinuität mithilfe der Zunahme-Matrix G und folgender Formel:

$$c_t(G) = \sum_{k=1}^{K} \frac{1}{\sigma_k^2} \sum_{t=2}^{T} (g_{k,t} - g_{k,t-1})^2 \tag{13}$$

Auch hier werden wieder große Abstände von aufeinanderfolgenden Einträgen mithilfe der Quadratfunktion geahndet und kleine Abstände fallen weniger ins Gewicht. Des Weiteren ist $\sigma_k^2 = (1/T) \sum_{t=1}^{T} g_{t_j}^2$ ein Mittelwert der quadrierten Einträge von g_k (Zeilenvektoren von G) und dient zur Normierung der Abstände. Ohne zusätzlicher Normierung könnte man sonst $c_t(G)$ kleiner werden lassen indem man einfach G durch einen möglichst großen Wert skalar teilt. Da die zeitliche Kontinuität von G dadurch aber logisch betrachtet eigentlich nicht zunimmt, wäre der Ansatz (13) ohne Normierung falsch.

Dünnbesetztheit

Eine Matrix ist genau dann dünn besetzt, wenn sie nur wenige Einträge ungleich 0 enthält. Für die Dünnbesetztheit der Spektrogramme der Quellsignale genügt es wieder, nur die Dünnbesetztheit von G zu untersuchen. Diese lässt sich durch folgende

Formel bemessen:

$$c_s(G) = \sum_{k=1}^{K} \sum_{t=1}^{T} f(\frac{g_{k,t}}{\sigma_k}) \tag{14}$$

Da man auch hier $c_s(G)$ durch skalare Division von G mit einem Wert x > 1 fäschlicherweise einfach minimieren könnte, ist die Normierung durch $\sigma_k = \sqrt{(1/T)\sum_{t=1}^{T} g_{t_j}^2}$ wieder wichtig. f ist eine beliebige Funktion, die Einträge ungleich 0 gewichtet. In der Implementierung von Virtanen und auch in dieser Arbeit wird $f(x) = |x|$ verwendet.

4.3 Der Lernalgorithmus

B und G werden am Anfang mit den absoluten Werten von Gausschem Rauschen initialisiert, das heißt jeder ihrer Einträge ist eine zufällig gewählte Größe aus der Standartnormalverteilung. Dann werden die Matrizen nach den Aktualisierungsregeln für die Basismatrix und die Zunahmematrix abwechselnd solange aktualisiert, bis die Kostenfunktion $c(B, G)$ (siehe (11)) konvergiert.

Aktualisierungsregel für die Basismatrix

Es wird die Aktualiserungsregel für B aus dem Algorithmus zur NMF mit der Kullback-Leibler-Divergenz übernommen.

$$B \leftarrow B. \times \frac{\frac{X}{BG}G^T}{\mathbf{1}G^T} \tag{15}$$

Herleitung einer Aktualisierungsregel für die Zunahmematrix

Um die Dünnbesetztheit und Kontinuität von G miteinfließen zu lassen, benutzt man ein Gradientenverfahren. Für den Gradienten einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$ gilt laut (3)

$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n, \quad (\nabla f) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Sei g_j die j-te Zeile von der Matrix G. Es gilt eigentlich:
$c_t : \mathbb{R}^{K \times T} \to \mathbb{R}$ (siehe (13)) und $c_s : \mathbb{R}^{K \times T} \to \mathbb{R}$ (siehe (14)). Man kann von $c_t(G)$ und $c_s(G)$ einen Gradienten bilden, indem man G zeilenweise beachtet.

Gradient für die Funktion der zeitlichen Kontinuität

Da wir die Kostenfunktion für die zeitliche Kontinuität zeilenweise betrachten gilt k=1, und somit:

$$c_t(g_j) = \sum_{k=1}^{K} \frac{1}{\sigma_k^2} \sum_{t=2}^{T} (g_{k,t} - g_{k,t-1})^2 = \frac{1}{\sigma_k^2} \sum_{t=2}^{T} (g_{k,t} - g_{k,t-1})^2 = \sum_{t=2}^{T} \frac{(g_{k,t} - g_{k,t-1})^2}{\frac{1}{T}\sum_{i=1}^{T} g_{k,i}^2} = \frac{\sum_{t=2}^{T} T(g_{k,t} - g_{k,t-1})^2}{\sum_{i=1}^{T} g_{k,i}^2}$$

Für $\frac{\partial c_t(G)}{\partial g_{k,t}}$ substituiere man nun, um die Quotientenregel $(\frac{u}{v})' = \frac{u'v - uv'}{v^2} = \frac{u'}{v} - \frac{uv'}{v^2}$ verwenden zu können.
Beachte außerdem die Kettenregel mit $u(v(g_{k,t})) = u'(v(g_{k,t}) \cdot v'(g_{k,t})$
und $\frac{\partial g_{i,j}}{\partial g_{k,t}} = \begin{cases} 1 & \text{falls } (i,j) = (k,t), \\ 0 & \text{sonst} \end{cases}$.
Nun fallen die meisten Variablen in der Summe weg, wodurch nur noch die Summanden abgeleitet werden müssen, die $g_{k,t}$ enthalten.
Durch die Substitution erhält man:

$$u(G) = \sum_{k=1}^{K} \sum_{t=2}^{T} T(g_{k,t} - g_{k,t-1})^2, \quad v(G) = \sum_{i=1}^{T} g_{k,i}^2$$

Für $1 < t < T$ gilt:
$\frac{\partial u(G)}{\partial g_{k,t}} = 2T(g_{k,t} - g_{k,t-1}) \cdot 1 + 2T(g_{k,t+1} - g_{k,t+1-1}) \cdot (-1) = 2T(2g_{k,t} - g_{k,t-1} - g_{k,t+1})$
Für t = 1 und t = T gilt:
$\frac{\partial u(G)}{\partial g_{k,1}} = 2T(g_{k,2} - g_{k,1}) \cdot (-1) = 2T(g_{k,1} - g_{k,2})$ $\quad \frac{\partial u(G)}{\partial g_{k,T}} = 2T(g_{k,T} - g_{k,T-1}) \cdot 1$
Die partielle Ableitung von $v(G)$ ist:
$\frac{\partial v(G)}{\partial g_{k,t}} = 2g_{k,t}$
Sei $[X]_{k,t}$ der Eintrag der Matrix X in der k-ten Zeile und t-ten Spalte. Somit ergibt sich gesamt für $1 < t < T$

$$\frac{\partial c_t(G)}{\partial g_{k,t}} = [\nabla c_t(G)]_{k,t} = 2T \frac{2g_{k,t} - g_{k,t-1} - g_{k,t+1}}{\sum_{i=1} g_{k,i}^2} - T \frac{2g_{k,t} \sum_{i=2}^{T} (g_{k,i} - g_{k,i-1})^2}{(\sum_{i=1}^{T} g_{k,i}^2)^2} = \frac{4Tg_{k,t}}{\sum_{i=1}^{T} g_{k,i}^2} - (2T \frac{g_{k,t-1} + g_{k,t+1}}{\sum_{i=1}^{T} g_{k,i}^2} + 2T \frac{g_{k,t} \sum_{i=2}^{T} (g_{k,i} - g_{k,i-1})^2}{(\sum_{i=1}^{T} g_{k,i}^2)^2}) = [\nabla c_t^+(G)]_{k,t} - [\nabla c_t^-(G)]_{k,t}$$

Für t = 1 gilt:

$$\frac{\partial c_t(G)}{\partial g_{k,1}} = [\nabla c_t(G)]_{k,1} = 2T\frac{g_{k,1} - g_{k,2}}{\sum_{i=1}^{T} g_{k,i}^2} - T\frac{2g_{k,1}\sum_{i=2}^{T}(g_{k,i} - g_{k,i-1})^2}{(\sum_{i=1}^{T} g_{k,i}^2)^2} =$$
$$\frac{2Tg_{k,1}}{\sum_{i=1}^{T} g_{k,i}^2} - (2T\frac{g_{k,2}}{\sum_{i=1}^{T} g_{k,i}^2} + 2T\frac{g_{k,1}\sum_{i=2}^{T}(g_{k,i} - g_{k,i-1})^2}{(\sum_{i=1}^{T} g_{k,i}^2)^2}) =$$
$$[\nabla c_t^+(G)]_{k,t} - [\nabla c_t^-(G)]_{k,t}$$

Für t = T gilt:

$$\frac{\partial c_t(G)}{\partial g_{k,T}} = [\nabla c_t(G)]_{k,T} = 2T\frac{g_{k,T} - g_{k,T-1}}{\sum_{i=1} g_{k,i}^2} - T\frac{2g_{k,T}\sum_{i=2}^{T}(g_{k,i} - g_{k,i-1})^2}{(\sum_{i=1}^{T} g_{k,i}^2)^2} =$$
$$\frac{2Tg_{k,T}}{\sum_{i=1}^{T} g_{k,i}^2} - (2T\frac{g_{k,T-1}}{\sum_{i=1}^{T} g_{k,i}^2} + 2T\frac{g_{k,T}\sum_{i=2}^{T}(g_{k,i} - g_{k,i-1})^2}{(\sum_{i=1}^{T} g_{k,i}^2)^2}) = [\nabla c_t^+(G)]_{k,t} - [\nabla c_t^-(G)]_{k,t}$$

Gradient für die Funktion der Dünnbesetztheit

Wir betrachten G wieder zeilenweise und erhalten:

$$c_s(g_j) = \sum_{t=1}^{T} |\frac{g_{k,t}}{\sigma_k}| = \frac{\sum_{t=1}^{T} g_{k,t}}{\sqrt{\frac{1}{T}\sum_{i=1}^{T} g_{k,i}^2}}$$

In obiger Gleichung entfällt der Betrag, weil die Matrix G nonnegativ ist und somit für deren Einträge $g_{k,t} > 0$ gilt.
Nun substituiere man wieder und erhält:

$$u(G) = \sum_{t=1}^{T} g_{k,t}, \quad v(G) = \sqrt{\frac{1}{T}\sum_{i=1}^{T} g_{k,i}^2}$$

Und es folgt:
$\frac{\partial u(G)}{\partial g_{k,t}} = 1$
$\frac{\partial v(G)}{\partial g_{k,t}} = \frac{1}{2\sqrt{\frac{1}{T}\sum_{i=1}^{T} g_{k,i}^2}} \cdot \frac{1}{T} 2g_{k,t} = \frac{\frac{1}{T}}{\frac{1}{\sqrt{T}}\sqrt{\sum_{i=1}^{T} g_{k,i}^2}} \cdot g_{k,t} = \frac{\frac{1}{\sqrt{T}} g_{k,t}}{\sqrt{\sum_{i=1}^{T} g_{k,i}^2}}$
Es ergibt sich gesamt:

$$\frac{\partial c_s(G)}{\partial g_{k,t}} = [\nabla c_s(G)]_{k,t} = \frac{1}{\sqrt{\frac{1}{T}\sum_{i=1}^{T} g_{k,i}^2}} - \frac{g_{k,t}\frac{1}{\sqrt{T}}\sum_{i=1}^{T} g_{k,i}}{\sqrt{\sum_{i=1}^{T} g_{k,i}^2} \cdot \frac{1}{T}\sum_{i=1}^{T} g_{k,i}^2} =$$
$$\frac{1}{\sqrt{\frac{1}{T}\sum_{i=1}^{T} g_{k,i}^2}} - \frac{g_{j,t}\sqrt{T}\sum_{i=1}^{T} g_{j,i}}{(\sum_{i=1}^{T} g_{j,i})^{3/2}} = [\nabla c_s^+(G)]_{k,t} - [\nabla c_s^-(G)]_{k,t}$$

Gradient für den Rekonstruktionsfehler

Es gilt

$$c_r(B,G) = \sum_{(f,t)\in F\times T} X(f,t) log \frac{X(f,t)}{BG(f,t)} - X(f,t) + BG(f,t) =$$
$$\sum_{(f,t)\in F\times T} X(f,t) log \frac{X(f,t)}{BG(f,t)} - \sum_{(f,t)\in F\times T} X(f,t) + \sum_{(f,t)\in F\times T} BG(f,t)$$

Der Gradient von $c_r(B,G)$ ist laut Lee und Seung [LS01] und Virtanen [Vir07], wenn wir G zeilenweise betrachten, gegeben durch:

$$\nabla c_r(B,G) = B^T(1 - \frac{X}{BG}) = B^T 1 - B^T \frac{X}{BG} = \nabla c_r^+(B,G) - \nabla c_r^-(B,G)$$

Aktualisierungsregel für die Zunahmematrix

Die Aktualisierungsregel ist dann schließlich wiefolgt:

$$G \leftarrow G. \times \frac{\nabla c^-(B,G)}{\nabla c^+(B,G)} \tag{16}$$

wobei

$$\nabla c^+(B,G) = \nabla c_r^+(B,G) + \alpha \nabla c_t^+(G) + \beta \nabla c_s^+(G) \tag{17}$$

und

$$\nabla c^-(B,G) = \nabla c_r^-(B,G) + \alpha \nabla c_t^-(G) + \beta \nabla c_s^-(G) \tag{18}$$

Hierbei ist zu beachten, dass durch die Aktualisierungsregel (16) die Kostenfunktion nicht zwingendermaßen konvergiert. Dies ist vor allem der Fall, wenn in (17) und (18) α zu hoch (100 <) gewählt wurde.

5 Implementierung der vorgestellten Verfahren

Um zu zeigen, dass Virtanen mit seinem in Kapitel 4 vorgestellten Ansatz tatsächlich ein funktionierendes Signaltrennungverfahren erschaffen hat, wird für diese Arbeit das Verfahren implementiert und getestet. Das dabei entstandene Programm wird auf den Namen „SOUNDSEPP“ getauft. Außerdem wird überprüft, ob es tatsächlich bereichsweise besser funktioniert als andere Verfahren. Dazu werden zwei Versionen der Nonnegativen Matrixfaktorisierung ohne zusätzliche Optimierungskriterien implementiert, zum Einen mit Kullback-Leibler-Divergenz als Kostenfunktion und zum Anderen mit Euklidischen Abstand als Kostenfunktion, beide gemäß den Alogrithmen in Kapitel 3.6.1. Des Weiteren soll die Auswirkung der Initialisierung der Startmatrizen auf das Ergebnis getestet werden, indem man für alle Algorithmen jeweils verschiedene Startmatrizen vorschlägt.

5.1 Implementierungsumgebung

- Als Entwicklungsumgebung wird die IDE „Octave“ der Version 5.1.0.0 verwendet und in der Programmiersprache Octave implementiert. Zum Ausführen und Testen des Codes auf relativ kurzen Audiodateien reicht die Rechenleistung eines Personal Computers. Deshalb wird für diese Arbeit ein Laptop unter Windows 10 mit 64-Bit-Architektur, 1,8GHz Quadcore-Prozessor und 8 GB RAM benutzt.

- Zur Implementierung der Signaltransformation und Visualisierung wird in großen Teilen auf die von Patricio López-Serrano, Christian Dittmar, Yiğitcan Özer und Meinard Müller entwickelte Toolbox „NMF Toolbox: Music Processing Applications of Nonnegative Matrix Factorization“ zurückgegriffen. Sie steht unter GNU-GPL Lizenz auf der Webseite der Audiolabs Erlangen zum Download zur Verfügung. Sie bietet unter anderem Audiodateien an und Matlab-Skripte an, mit denen verschiedene Formen der NMF getestet werden können. Die benötigten Skripte werden für diese Arbeit, wenn nötig, von Matlab in Octave übersetzt.

- Es müssen keine zusätzlichen Packages installiert werden.

5.2 Spezifikation der Funktionen

In der Architektur der zu implementierenden Software wird sich an das Model-View-Controller Prinzip gehalten. Die Funktionen sind in die Klassen „IO“ für Input-Output-Funktionen (View) und „Functions“ für Funktionen, die allein zur Implementierung des Algorithmus dienen (Model), eingeteilt. Den Part des Controller übernimmt eine Shell.

Funktionen ohne Klasse

Folgende Funktion wird implementiert:

- shell.m: Dies ist ein Skript für die Mensch-Maschine Interaktion. Sie bereitet die Arbeitsumgebung für den Algorithmus vor und führt die Query aus.

Input-Output

Folgende Funktionen werden aus der NMF-Toolbox entnommen:

- logFreqLogMag.m: Diese Funktion dient zum Skalieren des Magnituden-Spektrogramms vor der Visualisierung, sodass die Lautstärke in Dezibel angegeben werden kann und die Frequenz-Achse logarithmische Abstände hat.
- coloredComponents.m: Diese Funktion hilft, die einzelnen Quellsignal- Spektrogramme farbig unterschiedlich zu markieren und bei der Ausgabe der Signaltrennungsergebnisse innerhalb des Eingabespektrogramms einzufärben.
- visualizeComponentsNMF.m: Erstellt eine Grafik der Ergebnisse. Es werden Eingabespektrogramm (also X im Signalmodell $X \approx BG$), eine Visualisierung der Basisvektoren (der Matrix B) und eine Visualisierung der Zunahme der einzelnen Basisvektoren (der Matrix G) kombiniert.

Folgende Funktionen werden implementiert:

- query.m: Dieses Skript dient zur Abfrage aller benutzerdefinierten Parameter und übergibt die Parameter an die nötigen Funktionen zur Durchführung des Algorithmus.

Functions

Folgende Funktionen werden aus der NMF-Toolbox entnommen:

- forwardSTFT.m: Führt die Schnelle Fouriertransformation auf einem Signal aus und berechnet so das Spektrogramm des Signals.
- inverseSTFT.m: Verwandelt das Spektrogramm eines Signals wieder in das Signal zurück.

Folgende Funktionen werden implementiert:

- cost.m: Berechnet die Kostenfunktion. Man kann benutzerdefiniert zwischen den Algorithmen *KLDiv*, *EucDist* und *Vir* wählen, wobei *KLDiv* und *EucDist* die Algorithmen aus Kapitel 3.6.1 sind und *Vir* Virtanens Algorithmus aus Kapitel 4.

- grad.m: Berechnet den Gradienten der Zunahmematrix G des Signalmodells $X \approx BG$ unter Hinzunahme der zeitlichen Kontinuität und Dünnbesetztheit. Dieser wird für Virtanens Ansatz *Vir* benötigt.

- snr.m: Mit dieser Funktion kann man die Qualität der Signaltrennung messen, indem eine originale Quelldatei mit einer aus der Signaltrennung hervorgegangenen Quelldatei verglichen wird.

Folgende Funktionen werden aus der NMF-Toolbox übernommen, aber auf die Testabsichten dieser Arbeit angepasst:

- initActivations.m: Zum Erstellen der initialen Zunahmematrix G des Signalmodells $X \approx BG$. Es kann zwischen *uniform* (Initialisierung nur mit Einsen), *gaussian* (standartnormalverteilte Zufallszahlen zwischen 0 und 1) und *random* (gleichverteilte Zufallszahlen zwischen 0 und 1) gewählt werden.

- initTemplates.m: Zum Erstellen einer initialen Basismatrix B des Signalmodells $X \approx BG$. Es kann zwischen *uniform*, *gaussian* und *random* gewählt werden.

- NMF.m: Führt die Nonnegative Matrixfaktorisierung durch. Dabei wird zum Laufzeit einsparen nicht nach jeder Iteration die Kostenfunktion kontrolliert, sondern immer 50 Iterationen vor der Berechnung der Kostenfunktion ausgeführt.

5.3 Handbuch zur Verwendung des Programms SOUNDSEPP

Das für diese Arbeit erstellte Programm SOUNDSEPP kann über da Befehlsfenster der Octave GUI ausgeführt werden. Dabei ist zu beachten, dass im Dateibrowser der GUI der Ordner SOUNDSEPP geöffnet sein sollte. Man tippe den Befehl „shell“ im Befehlsfenster ein und bestätige die Eingabe mit Enter, dann erscheint eine Query, die der Reihe nach alle benutzerdefinierten Angaben abfragt. Beachte dabei, dass die Query bei falschen Eingaben sofort wieder von neu beginnt. Die Eingaben sollten auch genau so erfolgen wie angefragt, ohne Leerzeichen oder Fehler in der Groß- und Kleinschreibung, denn diese werden nicht automatisch erkannt. Die Ergebnisse werden im Ordner „SOUNDSEPP\output“ als Audiodatei gespeichert. Nachfolgend kann man in Abbildung 15 ein Beispiel für die Query sehen. Im Ordner „input“ stehen einige Testdateien zur Verfügung. Möchte man das Programm auf anderen Audiodateien ausprobieren, so sollte man diese im Ordner „SOUNDSEPP\input“ platzieren. Hat man angegeben, die Visualisierung des Ergebnisses sehen zu wollen, so erscheint die Grafik dann in einem extra Fenster, die zum Beispiel wie in Abbildung 15 aussieht:

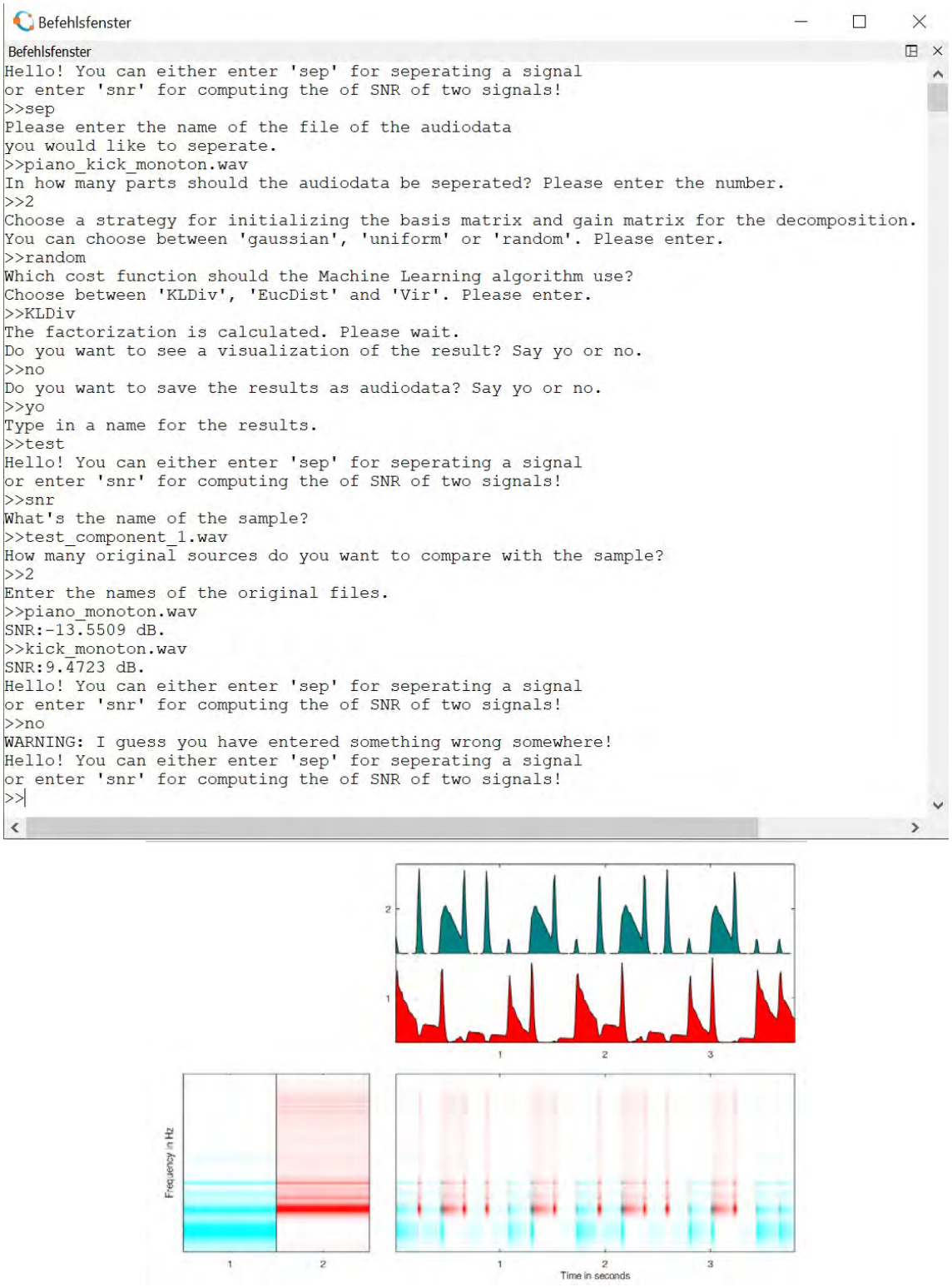

Abbildung 15: oben Befehlsfenster mit der Abfragequery, unten Visualisierung der Trennung der Audiodatei „SEPPsong\guitar_beats.wav“, der rote Bereich wird den Beats zugeordnet, der blaue Bereich der Gitarre

6 Auswertung des Verfahrens

6.1 Anforderungen an die Versuchsumgebung

Die klassischen Gütekriterien für wissenschaftliche Forschung sind Objektivität, Validität und Reliabilität. Auch die Auswertung der Trennungsergebnisse sollte diese Gütekriterien erfüllen.
Objektivität bedeutet, dass die Auswertungsergebnisse unabhängig von den Rahmenbedingungen, in denen man die Experimente durchführt, sind. Um diese zu erreichen, muss man einerseits ausschließen, dass die Auswertung und das Verfahren an sich Zufallskomponenten hat, die bei der mehrmaligen Durchführung dazu führen würden, dass dieselben Ausgangssignale unterschiedliche Ergebnisse und somit unterschiedliche Auswertungen hervorbringen. Dies ist in dem präsentierten Verfahren gegeben, da die einzige Zufallskomponente, und zwar die Initialisierung der Mischmatrizen, durch einen Seed immer die gleiche Zufallsmatrix liefert. Ein Seed ist ein Startwert für einen Pseudozufallszahlenalgorithmus. Weiterhin muss man beachten, dass die Auswahl der Messinstrumente keinen Einfluss auf die Ergebnisse hat. Auch dies ist natürlich gegeben, denn das Programm funktioniert auf jedem Rechner gleich.
Validität beurteilt, ob das gemessen wird, was man tatsächlich messen will. Mit der Signal-to-Noise-Ratio (SNR) kann man ein Orignialquellsignal mit dem zugehörigen aus der Signaltrennung hervorgegangenen Quellsignal vergleichen. Übertragen auf die Verwendung der SNR heißt Validität, dass man sicherstellen muss, die richtigen Signale miteinander zu vergleichen. Man könnte ein extra Zuordnungsverfahren implementieren, aber für eine kleine Stichprobe kann man auch manuell durch Hörtests entscheiden, welche Signale man miteinander vergleichen will. Oder man kann das getrennte Signal dem Originalsignal zuordnen, mit dem es den besten SNR-Wert besitzt.
Reliabiltiät ist ein Maß für die formale Genauigkeit des Messverfahrens. Sie ist also diejenige Variable, die in einem Experiment den Messfehler beziffert. Dieser existiert in diesem Fall nicht, denn ein durch den Computer entstandenes Produkt mit dem Computer zu messen, ist sehr exakt.

6.2 Signal-To-Noise-Ratio

Es besteht die Möglichkeit, Verzerrungen in Audiosignalen nochmal zu differenzieren und die SNR der einzelnen Verzerrungen zu berechnen. Eine mögliche Differenzierung wäre in Interferenzen, also Einwirkungen von anderen Quellkomponenten, Artefakte, also künstliche hervorgerufene Töne, und Rauschen. Für die folgende Evaluierung wird aber ein Ansatz gewählt, der die Verzerrung als gesamtes betrachtet bemisst.

Die SNR zweier Komponenten o (Original) und s (Quellsignal nach der Trennung) lässt sich berechnen, indem man die Spektrogramme X_o und X_s beider Signale vergleicht. Dazu folgende Formel:

$$SNR(o,s) = 10log_{10}(\frac{\sum_{f,t}[X_o]_{f,t}^2}{\sum_{f,t}([X_o]_{f,t} - [X_s]_{f,t})^2})dB$$

Es beschreibt das Verhältnis der mittleren Signallautstärke zur mittleren Störgeräuschlautstärke. Sie wird für gewöhnlich in Dezibel angegeben, deshalb verwendet man in der Formel den Logarithmus. Je höher die SNR, desto besser ist die Trennung. Ein positiver Wert der SNR bedeutet, dass das Signal über dem Störgeräusch überwiegt, ein negativer bedeutet, dass es mehr Störgeräusch als Signal gibt. Den der Logarithmus $log(x)$ ist für $x < 1$ negativ. Für $x \leq 0$ ist der Logarithmus nicht definiert, jedoch wird der Bruch durch die Quadrierungen nie negativ. Und solange das Orignialsignal nicht leer ist der Bruch auch nicht gleich 0.
Nonnegative Matrixfaktorisierung führt unverweigerlich dazu, dass kein einziger Einträg der resultierenden Spektrogramme gleich Null ist an Stellen an denen sie eigentlich Null sein sollten. Aber dadurch, dass diese eigentlich nicht hörbar sind, weil die Amplitude gegen Null geht und Werte nahe der Null den Betrag der SNR nicht stark beeinflussen, verringert sich die Qualität der NMF bemessen durch die SNR dadurch nur marginal. [VGF06]

6.3 Testdateien

Für die Generierung von Audiodateien zum Testen von SOUNDSEPP wird die Digital Audio Workstation (DAW) „FL Studio“ verwendet. Eine DAW ist ein Programm zur Tonaufnahme, Musikproduktion und Abmischung. Mit der kostenlosen Testversion der DAW „FL Studio“ kann man mithilfe von Samples und digitalen Synthesizern Musik komponieren und als Audiodatei speichern. Folgende Dateien stehen für die Tests zur Verfügung und sind auf beiliegendem Datenträger unter „SOUNDSEPP/input“ aufzufinden:

piano_kick_monoton.wav

Diese Audiodatei ist eine Kombination eines Pianos, dass monoton 3 mal regelmäßig hintereinander den gleichen Klang anspielt, während eine Kick-Drum je Klang 8 mal aufschlägt. Das Schlagen der Kick-Drum passiert ebenfalls in regelmäßigen Abständen. Zur Auswertung kann man die Dateien *piano_monoton.wav* und *kick_monoton.wav* heranziehen.

beats.wav

Diese Audiodatei besteht aus 3 Quellkomponenten nicht harmonischen Ursprungs. In nachfolgender Abbildung 16 sieht man eine Darstellung des Rhythmus, die nicht im klassischen Schlagzeugnotensystem dargestellt ist, da Schlagzeugnoten für den Laien schwer zu lesen sind. Die Tabelle ist von links nach rechts als zeitliche Abfolge zu lesen, das heißt mehrere Einträge pro Spalte deuten auf gleichzeitiges Auftreten der unterschiedlichen Instrumente in diesem Zeitpunkt hin.

Part 1:

Part 2:

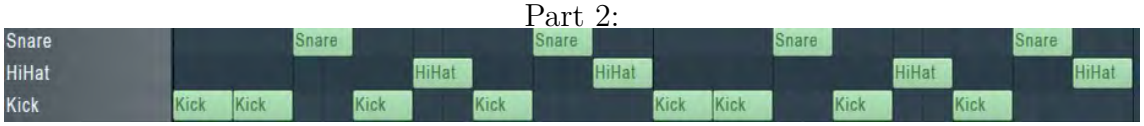

Abbildung 16: Tabelle für snare_kick_hihat

Auf beiliegendem Datenträger sind ebenso die drei Dateien *snare.wav*, *hihat.wav* und *kick.wav* einzeln zugänglich, die als originale Quellkomponenten zum Abgleich für die SNR dienen, wenn man nur die Datei *beats.wav* zerlegen möchte. Zusätzlich werden jeweils 2 Quelldateien kombiniert (*kick_snare.wav*, *hihat_snare.wav*, *kick_hihat.wav*) angeboten. Damit werden Vergleichsobjekte für den Versuch, zu testen ob eine Trennung in nur 2 Komponenten besser funktioniert, bereitgestellt. Dabei kann man beispielsweise *kick_snare.wav* selbst in 2 Komponenten trennen, oder beats.wav in 2 Komponenten trennen und validieren, ob nun 2 Ergebniskomponenten vorliegen, die beispielsweise *snare_kick.wav* und *hihat.wav* ähnlich sind.

piano.wav und guitar.wav

Bei *piano.wav* spielt ein Piano die C-Dur-Tonleiter, die bei C3 beginnt und bei C4 endet. Die Datei *guitar.wav* beinhaltet Klänge einer von einer Gitarre gespielten C4-Note, die vier mal langsam hintereinander angespielt wird und jeweils Zeit zum abklingen hat.

SEPPsong.wav

Diese Audiodatei ist eine Kombination von *beats.wav*, *piano.wav* und *guitar.wav*. Nachfolgend ein Hasse-Diagramm der Originalkomponenten, um den Zusammenhang noch einmal deutlich zu machen:

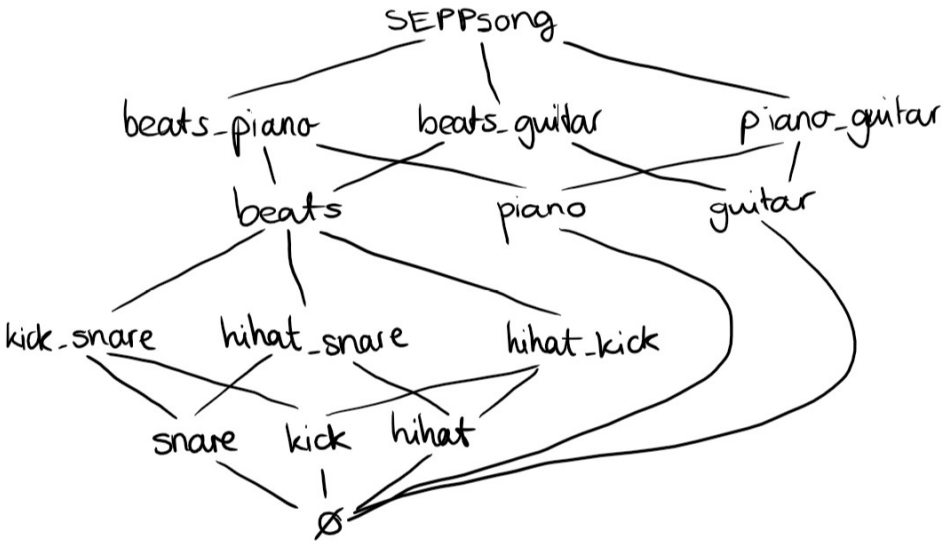

Abbildung 17: Hasse-Diagramm zur Datei SEPPsong

6.4 Auswertungsergebnisse

Im Folgenden werden die SNR und eine subjektive Einschätzung der Ergebnisse gegenübergestellt. Eine subjektive Einschätzung ist wichtig, da die Qualität der Audiosignaltrennung wesentlich auch von der Empfindung des Menschen abhängt. Sämtliche aus den Zerlegungstests hervorgegangenen Audidateien sind auf beiliegendem Datenträger im Ordner „Versuchsergebnisse“ aufzufinden.

Initialisierung 'uniform'

Wie in Kapitel 3.6.3 schon vorhergesagt, wird bei einer Initialisierung der Startmatrizen mit Einsen erreicht, dass das Signal stets in identische Teilsignale zerlegt wird. Beim Hörtest hören sich bei jedem Algorithmus (*EucDist, KLDiv und Vir*) Komponente 1 und Komponente 2 der Zerlegung genau so wie das Original an, sind aber etwas leiser. Auch die Komponenten der Algorithmen untereinander hören sich gleich an. Da man solches Resultat mit jeder Audiodatei erwarten kann, reicht an dieser Stelle ein einiziger Test mit *uniform* initialisierten Startmatrizen.
Jede Zerlegung in identische Teilmatrizen muss äquivalent zueinander sein. Die einfachste Zerlegung in identische Teilsignale ist in zwei Teilsignale mit der halben Amplitude des ursprünglichen Signals. Das erklärt, warum die Komponenten nach der Signaltrennung leiser als das Originalsignal sind. Es ist trotzdem interessant, dass alle drei Verfahren dieselben Ergebnisse liefern, denn die Basis- und Zunahme-Matrizen sind am Ende des Verfahrens unterschiedlich. Sichtbar wird dies z.B. in der folgenden Abbildung. Hier wurde die Audiodatei *piano_kick_monoton.wav* in 2 Komponenten mithilfe der 3 verschiedenen Algorithmen zerlegt und die Matrizen *uniform* initialisiert.

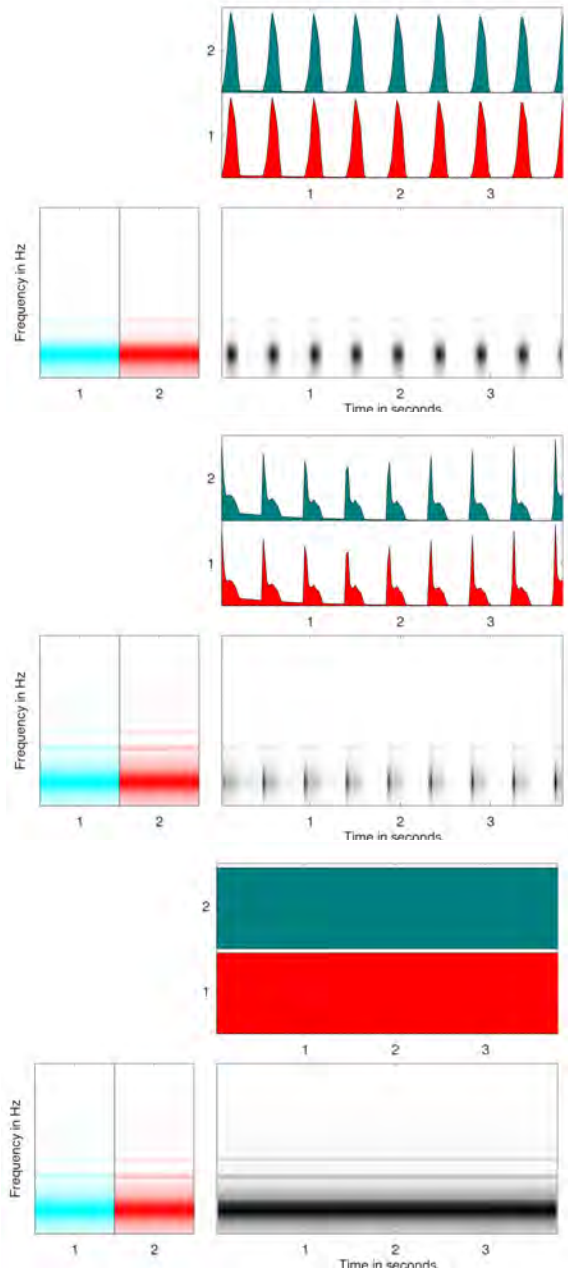

Abbildung 18: Oben wurde der *EucDist*-Algorithmus verwendet und in der Mitte *KLDiv*. Unten ist das Ergebnis des *Vir*-Algorithmus, bei dem die Dünnbesetztheit mit 0 und die zeitliche Kontinuität mit 100 gewichtet wurde. Dass der gesamte Bereich des Spektrogramms schwarz eingefärbt ist, deutet darauf hin, dass er keiner einzelnen Komponente zugeordnet werden konnte.

Gewichtung der zeitlichen Kontinuität und der Dünnbesetztheit beim *Vir*-Algorithmus

Der Algorithmus von Virtanen sieht vor, sowohl bei der Kostenfunktion als auch bei der Optimierung, die mithilfe des Gradients der Kostenfunktion erfolgt, eine Gewichtung vorzunehmen. Wie schon in Kapitel 4 besprochen, hat die Kostenfunktion folgende Form:

$$c(B, G) = c_r(B, G) + \alpha c_t(G) + \beta c_s(G)$$

α ist die Gewichtung der zeitlichen Kontinuität, β die Gewichtung der Dünnbesetztheit. Virtanen hat ohne Optimierungsverfahren verschiedene Gewichtungen durchprobiert und mit den Werten $\alpha = 100$ und $\beta = 0$ bei einer *gaussian* Initialisierung annähernd die besten Ergebnisse erzielt [Vir07]. Im Folgenden werden stichprobenartig Ergebnisse der Trennung von *piano_kick_monoton.wav* in 2 Komponenten mit anderer Gewichtung der zeitlichen Kontinuität und Dünnbesetztheit vorgestellt. Bei keiner der Zerlegungen ergibt eine Zuordnung von Ergebniskomponenten zu Originalkomponenten Sinn, weshalb keine SNR berechnet wird.

- Initialisierung *gaussian* mit $\alpha = 100$ und $\beta = 50$
  Die Datei wird in eine fast leere Komponente und eine Komponente zerteilt, die fast die gesamte Datei enthält. Die Vermutung liegt nahe, dass eine (fast) leere Komponente als optimales Ergebnis favorisiert wird, da sie natürlich die maximale Dünnbesetztheit besitzt. Treten in der fast leeren Komponente kurze Klänge auf, so klingen diese, als wäre der zu trennenden Datei einfach nur eine kurze Zeitsequenz entnommen worden.

- Initialisierung *random* mit $\alpha = 100$ und $\beta = 50$
  Hier gestaltet sich die Zerlegung so, dass eine der Komponenten relativ dünnbesetzt ist, jedoch in einem sehr unregelmäßigen Muster kurze Piano-Klänge und kurze Kickdrum-Klänge, sowohl einzeln als auch gemischt, enthält. Die andere Komponente ist dichtbesetzt und hört sich der zerlegten Datei noch immer sehr ähnlich an.

- Initialisierung *random* mit $\alpha = 0$ und $\beta = 100$
  Beide resultierende Komponenten sind nicht dünnbesetzt. Eine der Komponenten repräsentiert einen Teil der Frequenzen des Pianos mit Interferenzen, die durch die Kickdrum ausgelöst worden sind. Bei der anderen Komponente sind deutlich noch beide Quellkomponenten zu erkennen.

Für die folgenden Tests wird nun für den Algorithmus *Vir* stets $\alpha = 100$ und $\beta = 0$ gesetzt.

Welcher Algorithmus eignet sich kombiniert mit welcher Initialisierung für welchen Zweck am Besten?

Versuch 1: Für folgende Auswertung wird die Datei *piano_kick_monoton.wav* mithilfe verschiedener Algorithmen und verschiedener Initialisierungen in 2 Komponenten zerlegt. Die SNR der Datei *piano_mononton.wav* wird mit jeder der Ergebniskomponenten zusammen berechnet und die höchste SNR angegeben. Analog wird mit der Datei *kick_monoton.wav* verfahren. Ebenso wird darauf geachtet keine Ergebniskomponente mehreren Originalkomponenten zuzuordnen, auch wenn diese die höchste SNR besitzen. In diesem Fall wird die Kombination gewählt, die durchschnittlich die beste SNR aller Kombinationen besitzt.

SNR der Komponenten oben Piano unten Kickdrum	EucDist	KLDiv	Vir mit $\alpha = 100, \beta = 0$
Initialisierung *random*	- 8,59 dB 5,76 dB $\varnothing$ = - 1,41 dB	- 4,92 dB 9,47 dB $\varnothing$ = 2,27 dB	- 4,15 dB 9,91 dB $\varnothing$ = 2,88 dB
Initialisierung *gaussian*	- 8,57 dB 5,78 dB $\varnothing$ = - 1,39 dB	- 4,92 dB 9,47 dB $\varnothing$ = 2,27 dB	- 5,37 dB 8, 74 dB $\varnothing$ = 1,68 dB

Tabelle 1: Signaltrennung der Audiodatei *piano_kick_monoton.wav* in 2 Komponenten. *Vir* kombiniert mit *random* Intialisierung liefert hier das beste Ergebnis.

Die Werte der Tabelle sind auf 2 Nachkommastellen gerundet. Die Initialiserung mit KLDiv bringt bei einer *random* Initialisierung nicht exakt die gleichen SNR wie bei einer *gaussian* Initialisierung hervor, der Unterschied ist jedoch minimal.
Beim Hörtest erkennt man überraschenderweise keinen Unterschied der Zerlegung bezüglich der Initialisierung. Die Zerlegungen der verschiedenen Algorithmen hören sich sehr unterschiedlich an, jedoch ist je Algorithmus ein Unterschied der verschiedenen Initialisierungen nicht durch einen bloßen Hörtest erkennbar. Hervorzuheben ist, dass man bei den Hörtests klar erkennt, welche Komponente man der Kickdrum zuordnen kann und welche Komponente dem Piano.

Versuch 2: Gleicher Versuch wird mit der Audiodatei *SEPPsong/piano_guitar.wav* durchgeführt. Da beide original Quellkomponenten (*SEPPsong/guitar.wav* und *SEPPsong/piano.wav*) harmonisch sind, kann der Algorithmus eventuell Schwierigkeiten beim Zerlegen der Audiodatei haben, da sich im Spektrogramm nicht so leicht Muster erkennen lassen. Obwohl die Werte der folgenden Tabelle nicht schlecht erscheinen, ist beim Hörtest aller Ergebnisse in keinem einzigen Ergebnis eine klare Trennung von Piano und Gitarre zu erkennen.

SNR der Komponenten oben Piano unten Gitarre	EucDist	KLDiv	Vir mit $\alpha = 100, \beta = 0$
Initialisierung *random*	2,37 dB 5,04 dB $\varnothing = 3{,}70$ dB	3,70 dB 6,52 dB $\varnothing = 5{,}11$ dB	2,51 dB 4,97 dB $\varnothing = 3{,}74$ dB
Initialisierung *gaussian*	2,16 dB 2,86 dB $\varnothing = 2{,}51$ dB	3,85 dB 6,65 dB $\varnothing = 5{,}25$ dB	2,63 dB 5,27 dB $\varnothing = 3{,}95$ dB

Tabelle 2: Signaltrennung der Audiodatei *SEPPsong/piano_guitar.wav* in 2 Komponenten. *KLDiv* kombiniert mit *gaussian* Intialisierung liefert hier das beste Ergebnis.

Versuch 3: Da in vorherigen Versuchen einerseits eine Zerlegung einer Datei mit einer harmonischen und einer nicht harmonischen Quelle und andererseits eine Zerlegung einer Datei mit zwei harmonischen Quellen getestet wurde, soll folgender Test die Zerlegung in zwei nicht harmonische Quellen überprüfen. Dazu wird gleicher Versuch mit der Datei *SEPPsong/kick_snare.wav* durchgeführt.

SNR der Komponenten oben Kickdrum unten Snaredrum	EucDist	KLDiv	Vir mit $\alpha = 100, \beta = 0$
Initialisierung *random*	6,95 dB 4,50 dB $\varnothing = 5{,}72$ dB	6,27 dB 4,13 dB $\varnothing = 5{,}20$ dB	3,77 dB 1,55 dB $\varnothing = 2{,}66$ dB
Initialisierung *gaussian*	6,96 dB 4,50 dB $\varnothing = 5{,}73$ dB	6,27 dB 4,13 dB $\varnothing = 5{,}20$ dB	5,55 dB 3,28 dB $\varnothing = 4{,}41$ dB

Tabelle 3: Signaltrennung der Audiodatei *SEPPsong/kick_snare.wav* in 2 Komponenten. *EucDist* kombiniert mit *gaussian* Intialisierung liefert hier das beste Ergebnis.

Die Werte der Tabelle 3 könnten etwas verfälscht sein, bei diesem Test die Algorithmen *KLDiv* und *Vir* stets zum Abbruch gezwungen werden mussten, da die Kostenfunktion nach 1000 Iterationen noch nicht konvergiert hat. Jedoch ist auch mit einer lockereren Abbruchbedingung kein deutlich besseres Ergebnis zu erwarten. Beim Hörtest lässt sich gut, aber nicht sehr deutlich erahnen, welche Komponente man welcher Quelle zuordnen kann.

Fazit: Insgesamt lässt sich aus diesen drei Tests kein allgemeiner Favorit für alle Zwecke herausfinden. Jedoch ist bei diesen Tests besonders aufgefallen, dass eine Trennung in harmonisch und nicht harmonisch subjektiv betrachtet am besten funktioniert, da sich hier die gewünschte Trennung beim Hören am klarsten zeigt. Auch ist die Kickdrum der Kombination *Vir* mit *random* Initialisierung in Tabelle 1 das am besten herausgefilterte Instrument aller 3 Versuche.

Einfluss der Anzahl der Komponenten

Versuch 4: Im folgenden soll eine Zerlegung in 5 Komponenten mithilfe der Datei *SEPPsong/SEPPsong.wav* getestet werden. Erzielt man hier bessere SNR-Werte, so kann man eventuell einen Rückschluss von der Anzahl der Komponenten auf die Qualität der Trennung ziehen.

SNR der Komponenten Reihenfolge: Piano, Gitarre, Kickdrum, Snaredrum, HiHat	EucDist	KLDiv	Vir mit $\alpha = 100, \beta = 0$
Initialisierung *random*	- 1,51 dB 4,30 dB 3,63 dB 3,76 dB 1,06 dB $\varnothing$ = 2,25 dB	- 1,69 dB 5,85 dB 6,63 dB 3,33 dB 9,62 dB $\varnothing$ = 4,75 dB	0,56 dB 1,16 dB 2,99 dB 1,74 dB 1,01 dB $\varnothing$ = 1,49 dB
Initialisierung *gaussian*	- 3,63 dB 2,33 dB 5,10 dB 1,61 dB 2,95 dB $\varnothing$ = 1,67 dB	- 1,80 dB 4,45 dB 6,61 dB 4,14 dB 10,59 dB $\varnothing$ = 4,80 dB	0,36 dB 1,52 dB 4,71 dB 1,69 dB 2,63 dB $\varnothing$ = 2,18 dB

Tabelle 4: Signaltrennung der Audiodatei *SEPPsong/SEPPsong.wav* in 5 Komponenten. *KLDiv* kombiniert mit *gaussian* Intialisierung liefert hier das beste Ergebnis.

An diesem Versuch fällt besonders auf, dass der KLDiv-Algorithmus eine deutlich bessere durchschnittle SNR besitzt als die anderen Algorithmen. Allgemein ist hier aber kein positiver Einfluss der höheren Anzahl der Komponenten auf die SNR zu erkennen. Bei den meisten aus der Trennung hervorgegangen Dateien ist es schwierig, sie einer Quelle durch Hören zuzuordnen, da jetzt nicht nur eine Interferenz einer einzigen anderen Quelle auftreten kann, sondern Interferenzen von bis zu vier anderen Quellen. Somit kann man wegen der verwirrenden Interferenzen meist nicht mehr behaupten, eine Quelle besser herauszuhören als eine andere.

Fazit aus den Versuchen 1-4

In drei der vier Versuche schnitten die Algorithmen mit *gaussian* initialisierten Startmatritzen am Besten ab. In zwei Versuchen hat der Algorithmus *KLDiv* mit deutlichem Abstand die besten SNR-Ergebnisse hervorgebracht.
Haben die Algorithmen *EucDist* und *Vir* die besseren SNR-Ergebnisse, so nur mit geringem Vorsprung zu den anderen Algorithmen. Die Qualität hängt auch von der Wahl der Testdateien ab. Somit könnte es aufgrund der geringen Unterschiede bezüglich der SNR zu den anderen Algorithmen Zufall sein, dass diese Algorithmen

bei diesen Testaudiodateien am besten abschneiden.
Insgesamt liegt die Vermutung nahe, dass der *KLDiv*-Algorithmus kombiniert mit *gaussian*-Initialisierung auch bei weiteren Tests die besten Ergebnisse bemessen an der SNR hervorbringen könnte.

6.5 Vergleich mit Testergebnissen Anderer

Aus den Auswertungsergebnissen in vorigem Kapitel 6.4 lässt sich nicht eindeutig ableiten, welche Methode wann am Besten funktioniert. Das liegt an der relativ kleinen Anzahl an Tests. Im Rahmen dieser Arbeit, bei der es mehr darum geht die Algorithmen an sich zu verstehen als von den Algorithmen zu profitieren, ist ein groß angelegter Test aus Aufwandsgründen nicht in Frage gekommen. Um eine repräsentative Auswertung zu erhalten, die Hypothesen erlaubt, sind große Stichproben erforderlich, das heißt man braucht eine große Anzahl an Testaudiodateien. Außerdem muss eine automatisierte Testausführung implementiert werden und man benötigt ein erhöhtes Maß an Rechenleistung. Da es dennoch interessant ist, wie gut die Lernalgorithmen zur Signaltrennung funktionieren, werden im folgenden die Erkenntnisse aus [Vir07] und [MEKR11] über die Qualität der Trennung der in dieser Arbeit präsentierten Methoden stichpunktartig zusammengefasst:

- Die NMF eignet sich eindeutig besser als die Independent Subspace Analysis zum Finden von Quellsignalen.

- Bei der NMF werden bessere Ergebnisse erzielt, wenn die Kullback-Leibler Divergenz anstatt der Euklidischen Divergenz als Kostenfunktion verwendet wird.

- Die NMF unter Berücksichtigung der zeitlichen Kontinuität ist nur im Auffinden hoher Töne besser als die anderem Algorithmen. Allerdings ist zu beachten, dass die Gewichtung der zeitlichen Kontinuität auf keinen Fall zu hoch gewählt werden darf, denn dies hätte einen großen negativen Einfluss auf die Qualität der Ergebnisse oder führt eventuell dazu, dass der Algorithmus nicht terminiert. Berücksichtigt man die Dünnbesetztheit bei der NMF, führt das zu keiner Verbesserung der Signaltrennung, sondern meist zu einer Verschlechterung.

- Bei allen NMF-basierten Methoden gibt es umso weniger False Negatives (nicht identifizierte Quellsignale), je mehr tatsächliche Quellkomponenten die Testdatei hat. Zum Beispiel werden bei 10 Komponenten durchschnittlich 2 Komponenten nicht aufgefunden. Bei 20 Komponenten sind es im Verhältnis schon deutlich weniger, denn auch da sind es circa durschnittlich 2.

- Im Großen und Ganzen liefert die NMF mit den Kriterien zeitliche Kontinuität und Dünnbesetzheit bezüglich der SNR deutlich schlechtere Ergebnisse als die NMF mit der Kullback-Leibler Divergenz.

- Erwartet man von der Signaltrennung nur eine Trennung in harmonische und nicht harmonische Bestandteile, also genau 2 Komponenten die nicht zwingend Instrumente repräsentieren, erhält man meist zufriedenstellenden Ergebnisse.

- Die Wahl des Signaltrennungsalgorithmus soll von den Anforderungen abhängen. NMF hat sich als der beste Algorithmus zur Extrahierung von Schlaginstrumenten erwiesen. Aber um beispielsweise Melodien oder Bassinstrumente zu extrahieren, gibt es andere Algorithmen, die sich besser eignen.

7 Fazit

Die Interdisziplinarität des Themas Signaltrennung verleiht ihm sowohl seinen Reiz als auch seine Grenzen. Es eignet sich hervorragend als Beispiel dafür, warum es in Schule und Studium so wichtig ist, fachübergreifend zu unterrichten und ganzheitliches Denken, aber auch Computational Thinking zu fördern. Denn Signaltrennung wirft in allen MINT-Disziplinen Fragen auf:
Mathematik: Wie lässt sich das Problem der Komponentenfindung lösen?
Informatik: Wie repräsentiere ich Musik, sodass ich sie verarbeiten kann?
Naturwissenschaft: Wie funktioniert Hören?
Technik: Wie ist Musik beschaffen und digitalisierbar?

Diese Arbeit hat gezeigt, dass sich eine Zerlegung von Audiosignalen schon mit mit wenigen Mitteln umsetzen lässt und dabei oftmals schon zufriedenstellende Ergebnisse liefert. Jedoch ist in der Recherche zu dieser Arbeit auch deutlich geworden, dass bisherige Ansätze zur Audiosignaltrennung noch keine perfekten Ergebnisse liefern. Insbesondere das Fehlen einer Methode, die die Anzahl von Signalkomponenten verlässlich automatisch bestimmt, ist unzufriedenstellend. Es lohnt sich, die Forschung in der Signaltrennung von akustischen Signalen fortzuführen, denn der Einsatzbereich von Audiosignaltrennung ist groß und voller Marktlücken: automatische Instrumenterkennung, automatische Transkription von Audiodateien in Notenblätter, automatische Störgeräuschreduzierungen und vieles mehr. Dies sind alles Ideen, die seit langem existieren, aber noch keine perfekte Umsetzung besitzen. Es bleibt spannend, ob ein exakter Instrumenterkennungsalgorithmus bald dem menschlichen Spezialisten voraus sein kann – möglich wäre es.

Literaturverzeichnis

[Ack91] Ackermann, Philipp: *Computer und Musik*. Springer Vienna, 1991

[AHK+12] Arens, Tilo ; Hettlich, Frank ; Karpfinger, Christian ; Kockelhorn, Ulrich ; Stachel, Hellmuth ; Lichtenegger, Klaus: *Mathematik*. Spektrum Akademischer Verlag Heidelberg, 2012

[Ala08] Alapaydin, Ethem: *Maschinelles Lernen*. Oldenburg Wissenschaftsverlag, 2008

[DM16] Dittmar, Christian ; Müller, Meinard: Reverse Engineering the Amen Break – Score-informed Separation and Restoration applied to Drum Recordings. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24 (2016), Nr. 9, S. 1531–1543

[Dud] *Duden online*. `https://www.duden.de/rechtschreibung/Ton_Klang_Schwingung_Aufnahme`, `https://www.duden.de/rechtschreibung/Musik`. – Eingesehen am 25.3.2020

[KD06] Klapuri, Anssi ; Davy, Manuel: *Signal Processing Methods for Music Transcription*. Springer Science+Business Media LLC, 2006

[Lan] Lang, H.W.: *Schnelle Fouriertransformation*. `http://www.iti.fh-flensburg.de/lang/algorithmen/fft/fft.htm`. – Eingesehen am 04.01.2020

[LS01] Lee, Daniel D. ; Seung, Hyunjune Sebastian: Algorithms for non-negative matrix factorization. (2001)

[MEKR11] Müller, Meinard ; Ellis, Daniel ; Klapuri, Anssi ; Richard, Gaël: Signal Processing for Music Analysis. In: *Selected Topics in Signal Processing, IEEE Journal of Selected Topics in Signal Processing* Vol. 5 (2011), S. 1088 – 1110

[PKJ11] Puente León, Fernando ; Kiencke, Uwe ; Jäkel, Holger: *Signale und Systeme*. Oldenbourg, München, 2011

[Ran05] Randall, Robert H.: *An Introduction to Acoustics*. Dover Publications, 2005

[VGF06] Vincent, Emmanuel ; Gribonval, Rémi ; Fevotte, Cédric: Perfomance Measurement in Blind Audio Source Separation. (2006)

[Vir07] Virtanen, Tuomas: Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria. In: *IEEE Transactions on Audio, Speech and Language Processing* 15 (2007), Nr. 3, S. 1066–1074

[VVG18] Vincent, Emmanuel ; Virtanen, Tuomas ; Gannot, Sharon: *Audio Source Separation and Speech Enhancement*. John Wiley and Sons, 2018

[Woy19] Woyczynski, Wojbor A.: *A First Course in Statistics for Signal Analysis - Third Edition*. Birkhäuser, 2019

Abbildungsverzeichnis

Quellenangaben zu nicht selbst verfassten Abbildungen

2	Oszillogram vs Fourierspektrum	[Ack91, S. 10/15]
3	Aliasing	[Ack91, S.82]
4	Sägezahnfunktion	[Ack91, S.22]
5	Multiplikation mit der Rechteckfunktion	von Walter Dvorak via Wikimedia Commons (CC BY 3.0)
6	Leck-Effekt	[PKJ11, S.266]
7	Fensterfunktionen	von Bob K via Wikimedia Commons (gemeinfrei)
14	Beispiel zeitliche Kontinuität	[Vir07]

Tabellenverzeichnis

Eigenständigkeitserklärung

Hiermit bestätige ich Claudia Stockinger, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die wörtlich und sinngemäß übernommenen Passagen aus anderen Werken kenntlich gemacht habe. Die Arbeit ist weder von mir noch von einer anderen Person an der Universität Passau oder an einer anderen Hochschule zur Erlangung eines akademischen Grades bereits eingereicht worden.

Ort, Datum: Passau, 30.3.2020

Unterschrift: