Tontechnik 2 -
Dokumentation der Fallstudie

Vorgelegt am	01.08.2023
Name	Z. A. J. Asmara
Matrikelnummer	2416041
Gruppe	5
Studiengang	Medientechnik (BA)
Semester	SoSe23
Lehrveranstaltung	Tontechnik 2
Professorin	Prof. Dr. Eva Wilk
Hochschule	HAW Hamburg

Zia Asmara
2416041

Inhaltsverzeichnis

Zia Asmara
2416041

Vorwort

Die vorliegende Dokumentation verfasste ich im Rahmen meiner Prüfungsleistung "Tontechnik 2" im Ingenieur-Studium BA Medientechnik an der HAW Hamburg. Die Dokumentation umfasst vier praktische Probleme im Bereich Audio-Processing mit Python. Die ersten beiden Praxisprobleme habe ich eigenständig bearbeitet. Die restlichen Aufgaben ab PP3 habe ich gemeinsam mit Gruppe 5 durchgeführt. Die Zusammenarbeit verlief in der Regel so, dass wir die Aufgabenstellung und den Pseudocode in Python gemeinsam diskutierten und anschließend unseren eigenen Code erstellten, ohne die Aufgaben in spezifische Teile aufzuteilen.

Für die ganzen Python Programm wurde im Grunde vier Bibliotheken benutzt, nämlich:

1. Numpy (geschrieben np): für bearbeitung mit Nummer, Faltung
2. Soundfile (geschrieben sf): zum Aufrufen einer wav-Datei
3. Sounddevice (geschrieben sd): (optional) zum Abspielen eines wav-Datei
4. Matplotlib (geschrieben plt): zum Zeichnen die Plots / Diagrammen

Zia Asmara
2416041

PP1 - Nachhallzeit, Deutlichkeitsmaß und Klarheitsmaß

Das erste Praxisproblem besteht darin, die Nachhallzeit T, das Deutlichkeitsmaß C50 und das Klarheitsmaß C80 einer gegebenen Impulsantwort zu bestimmen. Als Teil von Gruppe 5 verwenden wir die Impulsantwort "Datei_C".

PP1 ist wichtig, um die Audio-Signalverarbeitung mit Python kennenzulernen, da ich damit noch keine Erfahrung habe. Die Bibliotheken, die ich brauchte, waren hauptsächlich numpy und soundfile, um die WAV-Daten zu lesen. Dann habe ich die Formeln gesammelt, um die Nachhallzeit, das Deutlichkeitsmaß und das Klarheitsmaß zu berechnen. Um den Umgang mit dem Signal zu lernen, habe ich gelernt, wie man Signale normalisiert, nur einen Kanal bei Stereo-Signalen verwendet und mit Arrays arbeitet.

Um die Nachhallzeit zu bestimmen, brauchen wir einen relativ linearen Abschnitt des Schroeder-Plots, der den 60dB-Energieabfall grafisch zeigt. Allerdings war dieser Bereich sehr klein, also habe ich mich entschieden, statt der Standard-Approximation T10 lieber T7 zu verwenden. Die Herausforderung zeigte sich durch den Korrelationsfaktor, der sich mit längerer Beobachtungszeit (T20, T30) von 1 (100% Übereinstimmung) unterscheidet.

Das Ergebnis für die Nachhallzeit T7 beträgt 1,93 Sekunden, das bedeutet, der Raum eignet sich nicht gut für Sprach- und Musikdarbietungen, es sei denn, er ist so groß wie ein Konzertsaal oder eine Kirche. Die berechneten Werte für das Deutlichkeitsmaß C50 und das Klarheitsmaß C80 bestätigen dies. Das Deutlichkeitsmaß ist -3,67 dB, das ist weniger als die Grenze von -3 dB für gute Sprachverständlichkeit. Das Klarheitsmaß ist -0,67 dB, unter der Grenze von 0 dB für hohe Transparenz, aber im Bereich von >= -1,6 dB, geeignet für klassische Musik.

Gemäß ISO 3382-1 zeigt die Abklingkurve, wie der Schalldruckpegel in einem Raum nach dem Abschalten der Schallquelle über die Zeit abnimmt. Die Abklingkurve kann nach dem eigentlichen Abschalten einer kontinuierlichen Schallquelle im Raum gemessen werden oder aus der rückwärts integrierten quadratischen Impulsantwort des Raums abgeleitet werden. Um die Ergebnisse zu veranschaulichen und ihre Plausibilität zu überprüfen, habe ich die Darstellung des Schroeder-Plots im Code beibehalten. Dieser zeigt die Zeit- und Energiebeziehung der Teilbereiche zur Gesamtenergie.

Zusammenfassend kann man sagen, dass die Nachhallzeiten in der WAV-Datei "Datei_C.wav" auf eine Raumakustik hindeuten, die eher trocken und hallarm ist, jedoch noch ausreichend Reflexionen aufweist, um ein angenehmes Klangumfeld zu schaffen. Die Sprachverständlichkeit könnte gut sein, da die Klarheitsmaße C50 und C80 relativ geringe negative Werte haben, aber dies hängt auch von anderen Faktoren wie der Anordnung der Schallquelle und der Zuhörerposition ab. Die Analyse der Nachhallzeiten bietet wichtige Informationen für die Beurteilung der Klangqualität und kann bei der Gestaltung und Optimierung der Raumakustik hilfreich sein.

Zia Asmara
2416041

PP2 - Panning Verfahren

Vor dem Programmieren war es wichtig, passende Testsignale auszuwählen. Ein Kriterium war, dass das Signal nicht zu kurz sein sollte, damit wir hören können, wie es sich von links nach rechts bewegt, und es sollten nur geringe Lautstärkeänderungen auftreten. Außerdem sollte es ein breitbandiges Signal sein, damit wir auch bei unbekannten Signalen die Richtung des Schalls richtig zuordnen können. Als erstes Testsignal wählte ich rosa Rauschen, auch als 1/f-Rauschen bekannt. Und als zweites Testsignal entschied ich mich für ein Musiksignal, wie z. B. Gitarre, um Vergleiche zu ermöglichen und das Hör- oder Testerlebnis interessanter zu machen.

Für die Verarbeitung in Python verwende ich dieselben Bibliotheken wie in PP1. Die Umwandlung des Signals von Mono zu Stereo (und umgekehrt) war für mich neu, aber auch eine sehr interessante Aufgabe. Für alle Aufgaben von A bis E versuche ich, ein Stereosignal in ein Monosignal umzuwandeln, indem ich beide Kanäle mit Python mitteln und dabei die signal.ndim-Eigenschaft verwende, die die Anzahl der Dimensionen im Signalarray zurückgibt. Wenn das Signal stereo ist, hat es zwei Dimensionen: eine für den linken Kanal und eine für den rechten Kanal. Ist es mono, hat es nur eine Dimension. Dann normalisiere ich das Signal auf einen Bereich von -1 bis 1. Für jede Teilaufgabe habe ich zwei Optionen hinzugefügt, um Audio 1 oder Audio 2 zu "abspielen", damit der Benutzer die Ergebnisse direkt vom Programm hören kann. Ich habe separate Funktionen für jede Teilaufgabe (teil_a bis teil_e) erstellt, um den Code leicht verständlich zu gestalten.

Die Funktion `teil_a` berechnet die Zeitdifferenz zwischen den linken und rechten Kanälen eines Audiosignals. Danach erzeugt sie ein Stereoton-Signal, indem sie eine Verzögerung im linken Kanal einführt, wodurch es beim Abspielen so klingt, als käme der Ton von rechts. Der linke Kanal wird um eine angegebene Zeitdifferenz verzögert, während der rechte Kanal unverändert bleibt. Das resultierende Stereoton-Signal wird mit der Funktion `abspielen` wiedergegeben.

Die Funktion `teil_b` soll ein Stereoton-Signal erzeugen, bei dem ein Unterschied in der Lautstärke zwischen den linken und rechten Kanälen besteht. Die Lautstärke des linken Kanals wird reduziert, wobei der Faktor auf 2,5 festgelegt wird, was einer Reduzierung um 75% entspricht (ungefähr -8 dB auf dem linken Kanal und 0 dB auf dem rechten Kanal). Dieser Faktor wurde basierend auf einer Dezibel-Berechnungsformel gewählt: `faktor = 10^(8/20)`, wobei 8 dB die gewünschte Reduzierung ist. Der rechte Kanal bleibt auf der Original-Lautstärke, wodurch das Audio beim Abspielen unausgewogen klingt.

Die Funktion `teil_c` zeigt die Beziehung zwischen teil a und teil b. Sie erzeugt ein Stereoton-Signal mit einer Verzögerung im linken Kanal und einer Lautstärkereduzierung im rechten Kanal. Beim Abspielen hört man eine leichte räumliche Verschiebung, als käme der Klang leicht später von links. Der rechte Kanal klingt deutlich leiser als der linke Kanal, wodurch ein ungleiches Gleichgewicht im Stereofeld entsteht. Dies simuliert die Wahrnehmung eines Audiosignals, das von einem Controller oder Mischer mit spezifischen Einstellungen für Verzögerung und Kanalabschwächung bearbeitet wird.

Teil D sucht nach der kleinsten Zeitverzögerung, bei der man zwei getrennte Signale (eines vom linken und eines vom rechten Kanal) im Stereoton-Signal wahrnehmen kann. Das interaktive Programm beginnt mit 5 ms Verzögerung und erhöht diese schrittweise bis maximal 45 ms. Der Benutzer wird gebeten, auf zwei getrennte Signale zu achten. Sobald man zwei separate Signale hört, stoppt das Programm und zeigt die Zeitverzögerung an. Dadurch kann man herausfinden, wie gut man zeitliche Unterschiede in Stereo-Audio wahrnehmen kann.

Bezüglich der in der Aufgabenstellung gegebenen Abbildungen konnte ich die Werte für Teil A und B bestätigen. Jedoch hatte ich bei Aufgabe C Schwierigkeiten, den dort beschriebenen Zusammenhang in PP2-1 vollständig zu bestätigen. Die Pegelunterschiede existieren, aber ich konnte das Signal nicht vollständig in der Mitte lokalisieren. Bei Aufgabe E konnte ich sie relativ leicht umsetzen, und das Signal scheint sich horizontal von links nach rechts zu bewegen.

Zia Asmara
2416041

PP3 - Verzerrung Analyse (mit Tanh) und Delay Effect (Echo und Reverse Echo)

Ich habe PP3 damit begonnen, zuerst an System B zu arbeiten, da ich bereits mit der Verzögerungsfunktion vertraut war und das Programm abschließen und in PP3 integrieren wollte. Anschließend habe ich mich mit der Untersuchung von Verzerrungsanalyse mittels "tanh" beschäftigt und das Programm für System A unter Verwendung der gleichen Bibliothek wie zuvor gestartet. Dieses Praxisproblem ist definitiv mein Favorit, da es unsere erste praktische Erfahrung in der Gruppe darstellt. Die produktiven Diskussionen mit Gruppe 5 haben mir unschätzbare Unterstützung und Einblicke geliefert. Sie haben mir geholfen, das Pseudocode zu erstellen und ein klares Verständnis dafür zu gewinnen, wie Verzerrungen mit "tanh" funktionieren und wie ich FFT nutzen kann, um den THD und den Klirrfaktor zu berechnen.

System A: Zuerst importiere ich die erforderlichen Bibliotheken für numerische Operationen, das Lesen/Schreiben von Audiodateien, die Audiowiedergabe und das Plotten. Die Hauptfunktion "tanh_verzerrung" nimmt ein Eingangssignal entgegen und wendet eine Verzerrung an, indem es das Signal normalisiert, die Tangenshyperbolicus-Funktion mit spezifizierten Parametern anwendet und dann das Ausgangssignal auf einen Arbeitspunkt auf der Kennlinie skaliert. Nachdem die ausgewählte Audiodatei geladen und vorverarbeitet wurde, berechne ich den Gewinn für die Verzerrungsfunktion und wende die Verzerrung auf das Eingangssignal an. Das verzerrte Ausgangssignal wird dann in einer neuen Datei namens "output.wav" gespeichert. Zusätzlich verwende ich die schnelle Fourier-Transformation (FFT), um den Frequenzinhalt sowohl des Eingangs- als auch des Ausgangssignals zu analysieren, was es mir ermöglicht, die Gesamt-Harmonische-Verzerrung (THD) und den Klirrfaktor des verarbeiteten Signals zu berechnen und auszugeben. Der Code bietet den Benutzern auch ein Menü, aus dem sie zwischen drei verfügbaren Audiodateien wählen oder ihre eigene Datei eingeben können, um ein flexibles und interaktives Erlebnis zu gewährleisten.

Nach Anwendung der Verzerrung mittels der Tangenshyperbolicus-Funktion (tanh-Funktion) sollte der Klang des Ausgangssignals im Vergleich zum ursprünglichen Eingangssignal reicher und harmonisch komplexer werden. Die tanh-Funktion fügt dem Signal Harmonische hinzu, indem sie die Eingangswerte komprimiert, was zur Verstärkung bestimmter Frequenzkomponenten führt. Dadurch entsteht ein warmer, gesättigter und leicht "dreckiger" Klang, der in der Audiobearbeitung oft als "Overdrive" oder "Soft Clipping" beschrieben wird. Die genauen Klangmerkmale und die Hörbarkeit der Harmonischen hängen von den gewählten Parametern (z. B. m und x_AP) und dem Inhalt des Eingangssignals ab. Niedrigere Werte von m erzeugen eine weichere Verzerrung mit weniger Harmonischen, während höhere Werte zu einem aggressiveren und verzerrten Klang mit stärker ausgeprägten Harmonischen führen.

System B: Die Funktion ist so erstellt, dass die den Echo-Effekt auf das Eingangssignal (input_signal_effekt) anwendet. Der Echo-Effekt erzeugt eine verzögerte Kopie des Eingangssignals und mischt es mit dem Originalsignal unter Verwendung eines Dämpfungsfaktors (decay). Dies erreiche ich, indem ich durch das Eingangssignal iteriere und ein Ausgangssignal erstelle, das die aktuelle

Eingabeprobe mit einer verzögerten und abgeschwächten Version von sich selbst kombiniert. Die Verzögerungszeit kann ich mit dem Parameter "delay" steuern und die Dämpfung mit dem Parameter "decay" anpassen. Das Ergebnis ist das Ausgangssignal "output_signal_effekt", das zurückgegeben wird.

Für den Reverse Echo-Effekt folge ich einem ähnlichen Prozess wie beim Echo-Effekt, jedoch füge ich das verzögerte und abgeschwächte Signal diesmal in umgekehrter Reihenfolge hinzu. Zuerst kehre ich das Eingangssignal mit np.flip() um und wende dann die Funktion "echo()" mit den angegebenen Verzögerungs- und Dämpfungsfaktoren auf das umgekehrte Eingangssignal an. Um den Reverse Echo-Effekt zu erhalten, kehre ich das resultierende Ausgangssignal erneut mit np.flip() um. Das Ausgangssignal "output_signal_effekt" wird dann zurückgegeben.

Nach Anwendung des Echo-Effekts wird die wahrgenommene Klangwiedergabe eine verzögerte Wiederholung des Originalsignals enthalten, was den Eindruck eines Echos erzeugt, das nach kurzer Zeit zurückkehrt. Die Charakteristika des Echos, wie seine Prominenz und das Ausblenden, werden durch die Verzögerungszeit und den Dämpfungsfaktor gesteuert. Andererseits erzeugt der Reverse Echo-Effekt eine wahrgenommene Klangwiedergabe als umgekehrte und verzögerte Wiederholung des Originalsignals. Dies erzeugt einen einzigartigen und markanten Effekt, bei dem der Klang mit Verzögerung rückwärts abgespielt zu werden scheint.

Zia Asmara
2416041

PP4 - Panning Pegeldifferenz-Verfahren und BRIR (Binaural Rooms Impulse Response)

Ich habe PP5 damit begonnen, sorgfältig geeignete Tracks auszuwählen, die als Grundlage für beide Teile a) und b) dienen und eine Vergleichbarkeit der verschiedenen Ansätze gewährleisten sollten. Mein Ziel war es, einen zusammenhängenden Mix zu erstellen, der einem Ausschnitt eines Songs ähnelt. Ich habe die folgenden 3 Komponenten ausgewählt: Drums-Snare, E-Gitarre und Lead-Vocals. Wenn sie gemeinsam gemischt werden, erzeugen diese Komponenten den Eindruck einer kleinen Band-Session in einem Studio.

Teil A:
Für Aufgabe A (Panning) habe ich jeden Track individuell mit der Panning-Technik auf der horizontalen Hörebene verschoben, bis der gewünschte Klang erreicht wurde. Ich habe den Code aus Praxisproblem 2 angepasst und auf jeden Track angewendet. Um Übersteuerungen zu vermeiden, habe ich die Tracks auf die gleiche Länge gesetzt und ihre Lautstärke auf -4,8 eingestellt, um eine passende Ausgabelautstärke zu gewährleisten. Jeder Track erhielt eine Zeitdifferenz von 0,2 ms. Die Tracks wurden dann zu einem Stereosignal gemischt, und ich habe das Signal-Array auf 20 Sekunden gekürzt. Das resultierende gemischte Signal wurde abgespielt und als output1.wav gespeichert.

Teil B:
Der Programmieraufwand für Teil B war ziemlich kompliziert, da ich viel Versuch und Irrtum mit der Faltung durchgeführt habe. Die Mono-Tracks und die BRIR mussten für die Faltung die gleiche Länge haben. Die Mono-Tracks wurden in Stereo mit beiden Stereo-Tracks der BRIR gefaltet. Zunächst habe ich versucht, beide Arrays mit der FFT vom Zeitbereich in den Frequenzbereich zu transformieren, aber der Faltungsprozess war sehr langsam. Stattdessen habe ich np.convolve verwendet, um den Prozess zu beschleunigen, aber es dauerte immer noch etwa 15 Minuten, bis das Ergebnis fertig war. Das resultierende Ausgangssignal von Teil b wurde als AKL_Output.wav gespeichert. Obwohl mein Code aufgrund der vielen Änderungen etwas redundant wurde, habe ich beschlossen, ihn beizubehalten, da er gut funktioniert, um Fehler zu vermeiden.

Ergebnisse:
Während der Mix von Teil A eher wie ein Studiomix klingt, erinnert der Mix von Teil B an eine Live-Veranstaltung/Aufnahme. Dies liegt daran, dass sich der Mix außerhalb des Kopfes befindet und einen "entfernteren" Klang hat, der an ein Konzert erinnert. Persönlich bevorzuge ich Mix A, da er intensiver und voluminöser klingt.

Literaturverzeichnis

Blauert, J., & Braasch, J. (2008). *Handbuch der Audiotechnik.* Springer Verlag.

Weinzierl, S. (2008). *Handbuch der Audiotechnik*, Chap. 5. Springer Verlag.

(kein Datum) *RT60 Reverberation Time*. Abgerufen am 14.06.2023 von https://svantek.com/academy/rt60-reverberation-time/

(kein Datum). *DSP Audio Effects Project*. Abgerufen am 25.07.2023 von https://kidpatel.wixsite.com/dspaudioeffects/distortion

Wilk, E. (2023). [Aufgabenstellungen zu allen Praxisproblemen]. HAW HH-DMI-MT.

Zia Asmara
2416041

Anhang

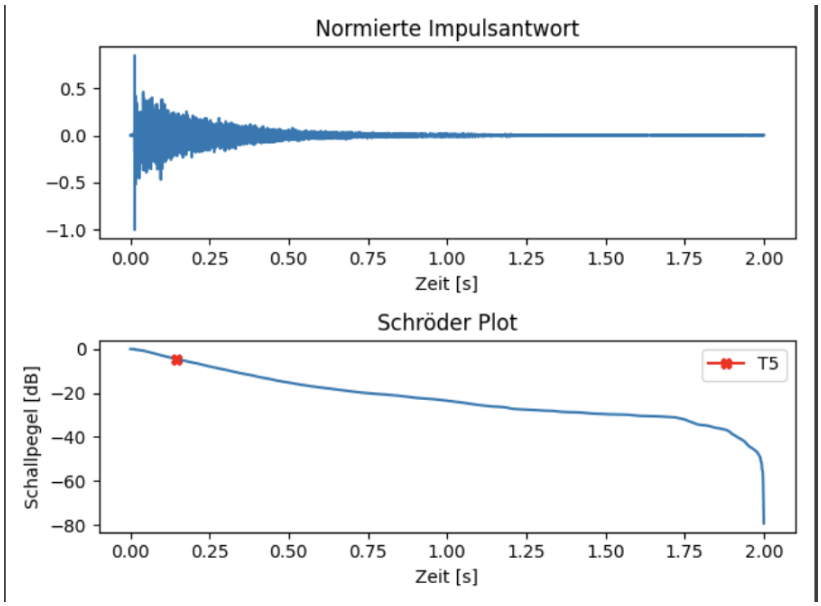

Abb.1. PP1 - Schroeder Plot

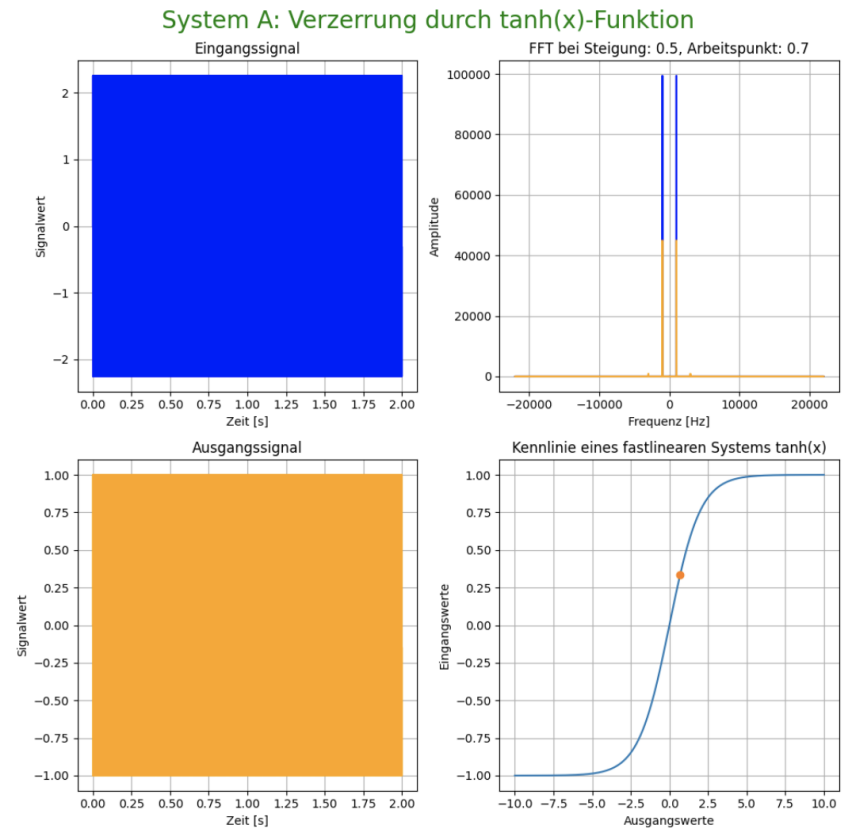

Abb.2. PP3 - Verzerrung durch tanh Funktion