Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Signale, Systeme und Sensoren

Aufbau eines einfachen Spracherkenners

D. Wollman, V. Bratulescu

Konstanz, 10. Januar 2021

Zusammenfassung (Abstract)

Thema:	Aufbau eines einfachen Spracherkenners	
Autoren:	D. Wollman	da161wol@htwg-konstanz.de
	V. Bratulescu	vl161bra@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz	mfranz@htwg-konstanz.de
	Jürgen Keppler	juergen.keppler@htwg-konstanz.de
	Mert Zeybek	me431zey@htwg-konstanz.de

In dem vierten Versuch geht es um den Aufbau eines einfachen Sprachenerkenners und dessen Auswertung. Der Spracherkenner soll simple Befehle wie 'Hoch', 'Tief', 'Links' und 'Rechts' erkennen. Der Aufbau soll nach dem Prinzip des Prototyp-Klassifikators geschehen. Die Spektren werden mithilfe der Windowing-Methode berechnet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Kapitel 1

Einleitung

In dem vierten Versuch geht es um den Aufbau eines einfachen Sprachenerkenners und dessen Auswertung. Der Spracherkenner soll simple Befehle wie 'Hoch', 'Tief', 'Links' und 'Rechts' erkennen. Der Aufbau soll nach dem Prinzip des Prototyp-Klassifikators geschehen. Die Spektren werden mithilfe der Windowing-Methode berechnet.

Für die Aufnahme wird ein Mikrofon verwendet, dass mit der Soundkarte des Computers verbunden ist und mithilfe der pyaudio Bibliothek aufgenommen wird. Im python Programm wird zusätzlich eine Triggerfunktion hinzugefügt, damit alle Signale an dem selben Zeitpunkt anfangen. Außerdem werden die Korrelationskoeffizienten nach Bravais-Pearson zum Vergleich zweier Eingabespektren verwendet.

Kapitel 2

Versuch 1: Fourieranalyse lang andauernder Signale

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

In dem ersten Teil des Versuchs soll ein Beispielsignal mithilfe von pyaudio aufgenommen werden. Dazu soll eine Triggerfunktion implementiert werden, die das Signal auf eine Sekunde kürzt und nur den relevanten Teil des Signals betrachtet, der einen gewissen Schwellenwert überschritten hat. Im Anschluss daran soll mithilfe von Windowing das Amplitudenspektrum berechnet werden.

Das Signal wird über ein Mikrofon mittels der pyaudio Bibliothek aufgenommen und anschließend in einer .csv Datei abgespeichert. Das verwendete Mikrofon wurde in der Nähe des Sprechers platziert, sodass das gesprochene Wort gut aufgenommen werden kann. Das Mikrofon wird über USB direkt mit dem Computer verbunden. Das aufzunehmende Signal soll ein beliebiges Wort sein und ein einem Diagramm dargestellt werden.

Durch den Code aus dem dritten Versuch können wir das Amplitudenspektrum der Aufnahme bestimmen und graphisch darstellen. Dazu wird die Windowing Methode hinzugefügt. Die Länge eines einzelnen Windows soll 512 Samples betragen und die Windows untereinander sollen sich bis zur Hälfte überlappen. Die einzelnen Windows werden mit einer Gaußschen Fensterfunktion multipliziert, welche eine Standardabweichung von 4 hat. Dabei wird in jedem Window eine lokale Fouriertransformation durchgeführt. Die Fouriertransformierte wird dann über alle Fenster gemittelt und daraus das Spektrum berechnet. Zum Schluss soll dieses mit dem eingangs errechneten Spektrum verglichen und auf Korrektheit überprüft werden.

2.2 Messwerte

In Abbildung 2.1 wird die Sprachaufnahme des Wortes ’Hallo’ dargestellt.

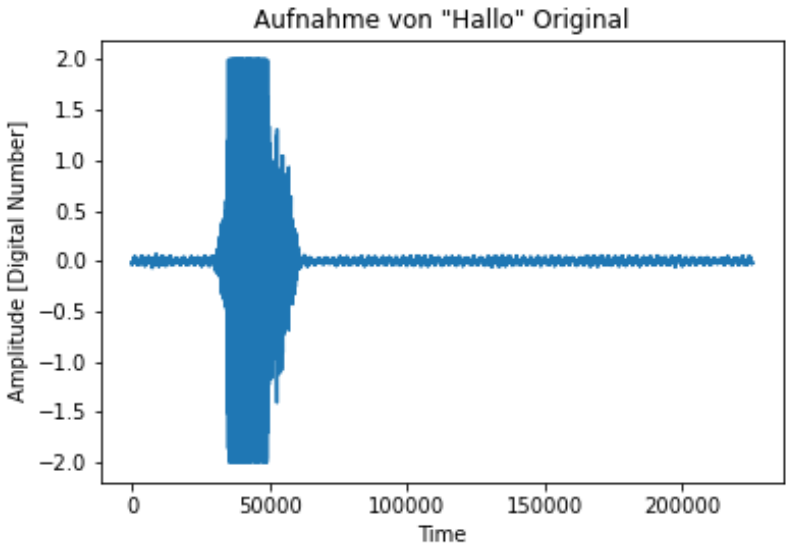


Abbildung 2.1: Sprachaufnahme des Wortes ’Hallo’

In Abbildung 2.2 wird die Triggerfunktion auf das obere Signal angewendet und mit Nullen aufgefüllt.

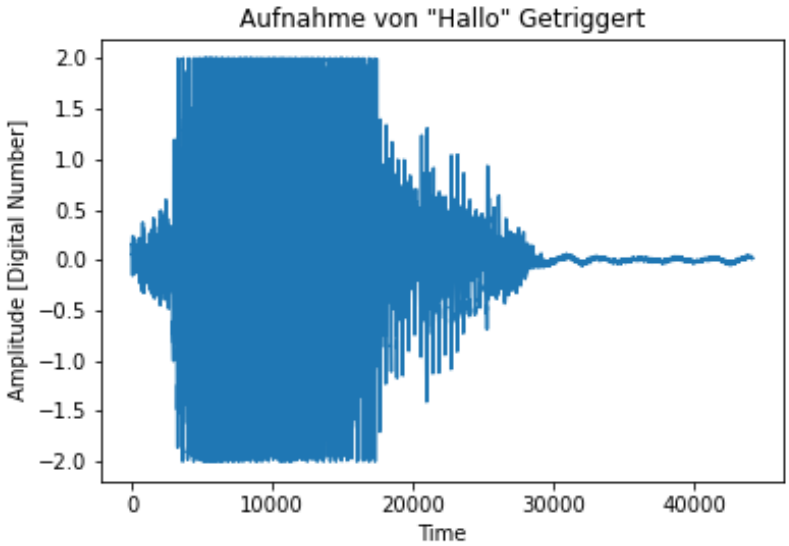


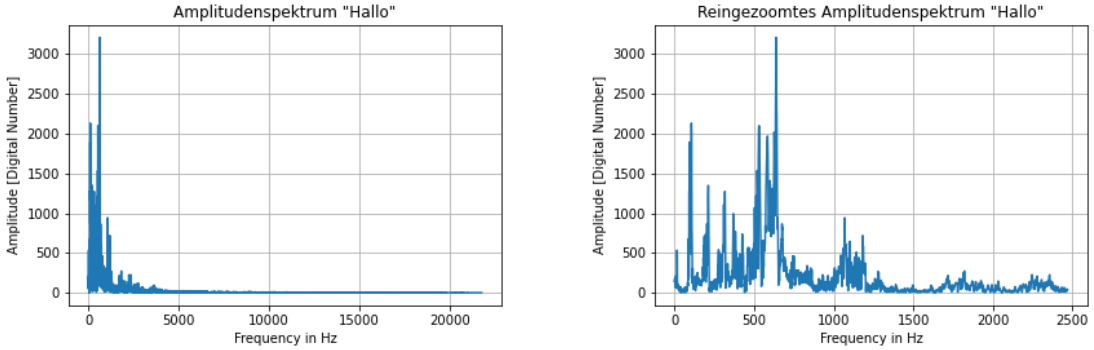
Abbildung 2.2: Sprachaufnahme des Wortes ’Hallo’ mit Triggerfunktion

2.3 Auswertung

Mit der Formel

$$f = \frac{n}{M \cdot \Delta t}$$

kann das errechnete Spektrum im Frequenzbereich dargestellt werden. Auf der linken Seite ist das gesamte Amplitudenspektrum zu sehen und auf der rechten Seite nur einen Ausschnitt aus 2500 Samples.



(a) Amplitudenspektrum der Sprachaufnahme (b) Ausschnitt des Amplitudenspektrums

Abbildung 2.3: Das Amplitudenspektrum mit der dazugehörigen Frequenz

In Abbildung 2.4 ist die verwendete Fensterfunktion mit einer Standardabweichung von 4 dargestellt.

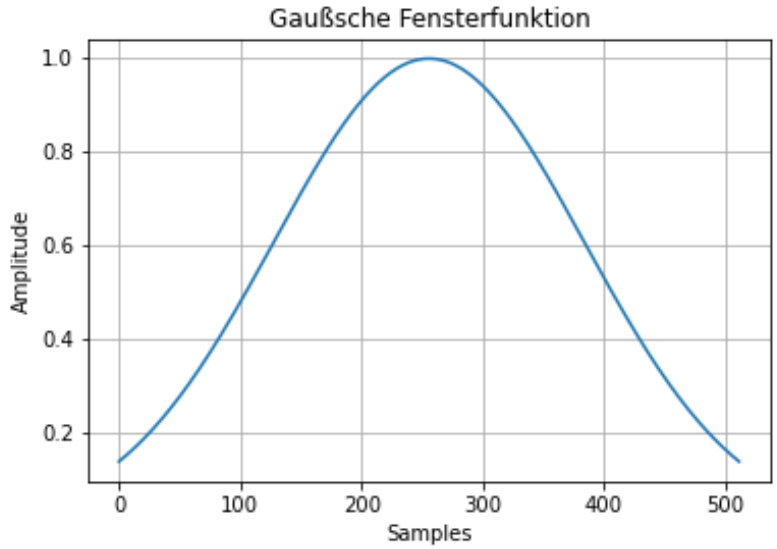
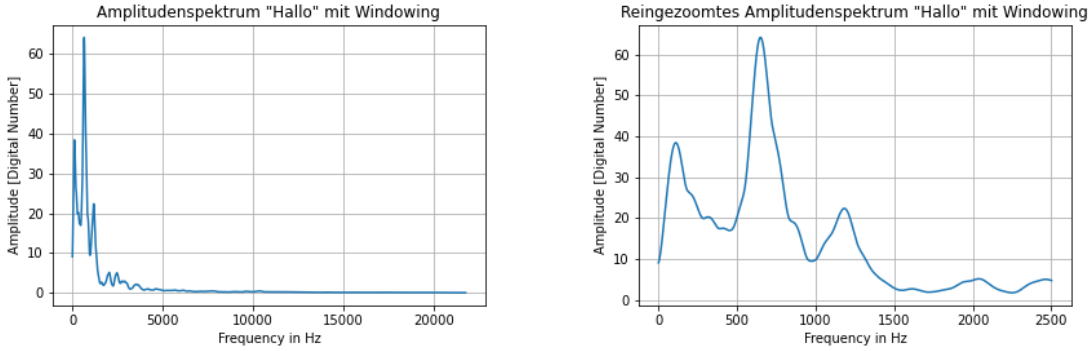


Abbildung 2.4: Gaussche Fensterfunktion mit Standardabweichung 4

In Abbildung 2.5 ist das durch Windowing errechnete Amplitudenspektrum zu sehen.



(a) Amplitudenspektrum der Sprachaufnahme mit Windowing

(b) Ausschnitt des Amplitudenspektrums mit Windowing

Abbildung 2.5: Das Amplitudenspektrum mit der dazugehörigen Frequenz

2.4 Interpretation

Im Vergleich von Abbildung 2.1 und 2.2 ist deutlich zu erkennen, dass der Trigger erst ab dem von uns definierten Schwellenwert ausgelöst wird und 44100 Samples lang dauert, was genau einer Sekunde entspricht.

Im Amplitundenspektrum kann man erkennen, dass es sich hierbei um kein periodisches Signal handelt. Außerdem wird dieses auf die Hälfte gekürzt, da sich das Spektrum ab der Hälfte wierderholt.

In Abbildung 2.5 kann man schön sehen, dass das errechnete Amplitudenspektrum durch Windowing sehr ähnlich mit dem Spektrum des Originalsignals ist.

Kapitel 3

Versuch 2: Spracherkennung

3.1 Fragestellung, Messprinzip, Aufbau, Messmittel

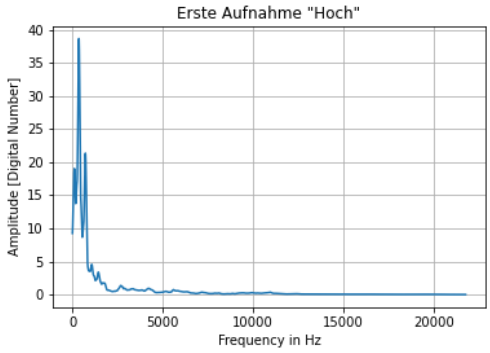
Im zweiten Teil des Versuches sollen die Befehle 'Hoch', 'Tief', 'Links' und 'Rechts' jeweils fünf Mal aufgenommen und jeweils das Spektrum berechnet werden. Anschließend soll aus den einzelnen fünf Aufnahmen, ein Referenzsprektrum berechnet werden. Dabei ist darauf zu achten, dass alle Aufnahmen vom gleichen Sprecher gesprochen werden.

Danach werden jeweils fünf Mal die Befehle von zwei unterschiedlichen Personen aufgenommen. Im Anschluss wird von jedem Testdatensatz das Spektrum berechnet und mit den Referenzsprektren verglichen. Als Maßtab wird der Korrelationskoeffizient nach Bravais-Pearson verwendet. Umso näher der Wert bei 1 ist, desto größer ist die Ähnlichkeit zu dem Referenzspektrum.

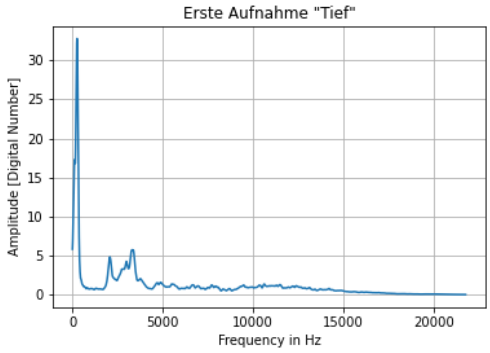
Zum Schluss soll noch die Detektions- und Fehlerrate angegeben werden.

3.2 Messwerte

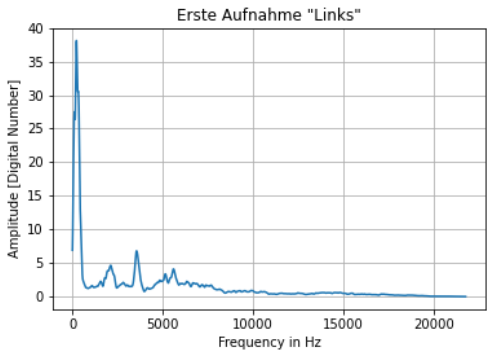
In Abbildung 3.1 werden die Spektren der ersten Aufnahmen der jeweiligen Befehle dargestellt. Die Spektren wurden ebenfalls durch das Windowing berechnet.



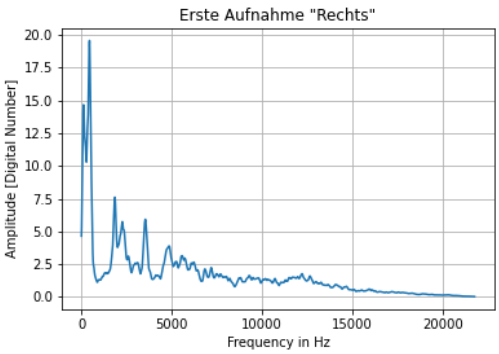
(a) Spektrum vom Befehl "Hoch"



(b) Spektrum vom Befehl "Tief"



(c) Spektrum vom Befehl "Links"

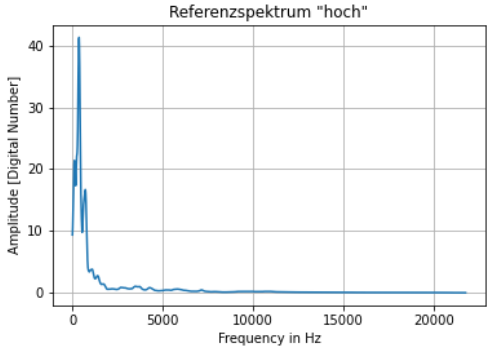


(d) Spektrum vom Befehl "Rechts"

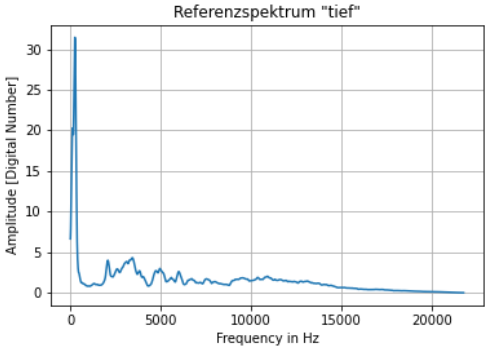
Abbildung 3.1: Die Spektren der ersten Aufnahmen

3.3 Auswertung

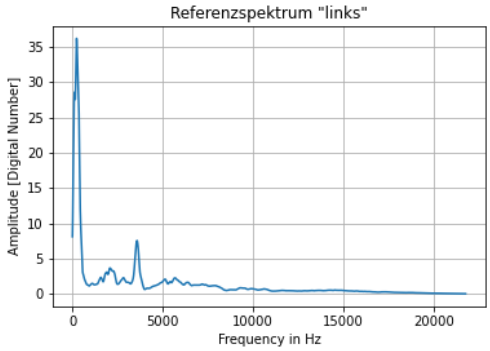
In Abbildung 3.2 werden die Referenzsprektren der vier Befehle dargestellt. Das Referenzspektrum haben wir wie folgt berechnet: Zuerst haben wir von jedem der fünf Aufnahmen pro Befehl, das Spektrum mithilfe von Windowing berechnet und anschließend aus den fünf Spektren den Mittelwert daraus genommen.



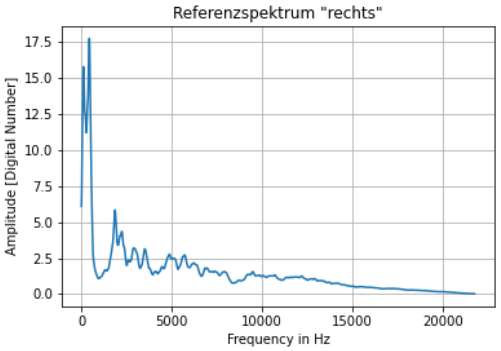
(a) Referenzspektrum vom Befehl "Hoch"



(b) Referenzspektrum vom Befehl "Tief"



(c) Referenzspektrum vom Befehl "Links"



(d) Referenzspektrum vom Befehl "Rechts"

Abbildung 3.2: Die Referenzsprektren der Befehle

In den nachffolgenden Tabellen sind die Testdatensätze zu sehen, die mit dem Referenzspektren verglichen wurden. In der linken Spalte ist der Name des Testdatensatzes beschrieben. In der zweiten Spalte steht der Korrelationskoeffiezient zu dem erkannten Befehl. Der erkannte Befehl befindet sich in der dritten Spalte. Ganz rechts in der Tabelle steht, ob die Schätzung des Python Programms korrekt war.

In Tabelle 3.1 sind die Ergebnisse von Person 1 (Sprecher der Referenzspektren) ersichtlich.

Befehl	**Korrelationskoeffizient**	**Geschätzt**	**Erkannt?**
hoch1_training.csv	0.6392205098106617	Hoch	Ja
hoch2_training.csv	0.6908852507975607	Hoch	Ja
hoch3_training.csv	0.7369898355976126	Hoch	Ja
hoch4_training.csv	0.6531021613772101	Hoch	Ja
hoch5_training.csv	0.5846061001105898	Hoch	Ja
tief1_training.csv	0.6917387323004587	Tief	Ja
tief2_training.csv	0.597228535138889	Tief	Ja
tief3_training.csv	0.5857716921817915	Tief	Ja
tief4_training.csv	0.5184803106129552	Tief	Ja
tief5_training.csv	0.4920849692590793	Tief	Ja
links1_training.csv	0.683875009035533	Links	Ja
links2_training.csv	0.6877934469896636	Links	Ja
links3_training.csv	0.6764595830513251	Links	Ja
links4_training.csv	0.7451023996899163	Links	Ja
links5_training.csv	0.6901293090260275	Links	Ja
rechts1_training.csv	0.7439299919720688	Rechts	Ja
rechts2_training.csv	0.6660542874986765	Rechts	Ja
rechts3_training.csv	0.5576880105772456	Rechts	Ja
rechts4_training.csv	0.6184389154587102	Rechts	Ja
rechts5_training.csv	0.6147002078697453	Rechts	Ja

Tabelle 3.1: Korrelationskoeffizienten von Person 1 bzw. des Spechers der Referenzspektren

In Tabelle 3.2 sind die Ergebnisse von Person 2 zu sehen.

Befehl	Korrelationskoeffizient	Geschätzt	Erkannt?
hoch1_training.csv	0.5617656486091793	Hoch	Ja
hoch2_training.csv	0.6034073877089225	Hoch	Ja
hoch3_training.csv	0.6609418303129206	Hoch	Ja
hoch4_training.csv	0.5627980823380955	Hoch	Ja
hoch5_training.csv	0.60474571557112	Hoch	Ja
tief1_training.csv	0.796451643049229	Links	Nein
tief2_training.csv	0.7883298868956207	Links	Nein
tief3_training.csv	0.7931331080053363	Links	Nein
tief4_training.csv	0.784277955016512	Links	Nein
tief5_training.csv	0.7574809465314387	Links	Nein
links1_training.csv	0.5950941597762736	Rechts	Nein
links2_training.csv	0.5975798374627092	Rechts	Nein
links3_training.csv	0.6181404787738591	Rechts	Nein
links4_training.csv	0.653205184868384	Rechts	Nein
links5_training.csv	0.637358727357185	Rechts	Nein
rechts1_training.csv	0.5274964804138591	Rechts	Ja
rechts2_training.csv	0.5182675549402136	Rechts	Ja
rechts3_training.csv	0.4888507104169534	Rechts	Ja
rechts4_training.csv	0.46684747064840704	Rechts	Ja
rechts5_training.csv	0.47841493852970246	Rechts	Ja

Tabelle 3.2: Korrelationskoeffizienten von Person 2

Person	Detektionsrate	Fehlerrate
Person 1	100%	0%
Person 2	50%	50%

Tabelle 3.3: Detektions- und Fehlerrate der Personen

3.4 Interpretation

Die Ergebnisse von Person 1, welcher der Sprecher der Referenzspektren war, sind wie zu erwarten, mit einer Detektionsrate von 100% einwandfrei. Jedoch sieht das bei Person 2 anders aus, da hier die Detektionsrate nur bei 50% liegt. Beispielweise wurde anstatt 'Tief' der Befehl 'Links' erkannt, was sich darauf zurückzuführen lässt, dass die Referenzspektren von 'Tief' und 'Links' in den niedriegen Frequenzen sehr identisch sind und dadurch stark korrelieren.

Schlussfolgernd lässt sich die Erkenntiss schließen, dass wenn man sich ein Spracherkenner für den privaten Gebrauch bauen möchte, dies eine gute Methode ist und sehenswerte Ergebnisse liefert. Jedoch ist diese Methode nicht für die breite Masse geeignet, da in unserem Beispiel eine Detektionsrate von 50% zu ungenau ist, wenn andere Stimmen mit ins Spiel kommen. Eventuell hätte der Spracherkenner bessere Ergebnisse geliefert, wenn wir für die Berechnung der Referenzspektren noch mehr Testdatensätze verwendet hätten, wie auch verschiedene Sprecher.

Anhang

A.1 Quellcode

A.1.1 Quellcode Versuch 1

```
import time
import pyaudio
import numpy
import matplotlib.pyplot as plt
FORMAT = pyaudio.paInt16
SAMPLEFREQ = 44100
FRAMESIZE = 1024
NOFFRAMES = 220

start = time.time()

p = pyaudio.PyAudio()
print("running")
stream = p.open(format=FORMAT,channels=1,rate=SAMPLEFREQ,input=True,frames_per_buffer=FRAMESIZE)
data = stream.read(NOFFRAMES*FRAMESIZE)
decoded = numpy.fromstring(data, "Int16");
stream.stop_stream()
stream.close()
p.terminate()

end = time.time()

abtastintervall = ((end - start) * 1000) / (NOFFRAMES*FRAMESIZE)

plt.plot(decoded)
plt.title('Aufnahme von "Hallo"')
plt.xlabel('Time')
plt.ylabel('Amplitude [Digital Number]')
```

```
plt.savefig('AufnahmeHallo.png')
plt.show()

decodedWithTimestamp = numpy.zeros((decoded.size,2))

for i in range(decoded.size):
    decodedWithTimestamp[i,0] = decoded[i]
    decodedWithTimestamp[i,1] = abtastintervall * (i)

numpy.savetxt("hallo.csv", decodedWithTimestamp, delimiter=",")
```

Listing 4.1: Skript Versuch 1a

```
import pyaudio
import numpy as np
import matplotlib.pyplot as plt
FORMAT = pyaudio.paInt16
SAMPLEFREQ = 44100
FRAMESIZE = 1024
NOFFRAMES = 220

p = pyaudio.PyAudio()
print("running")
stream = p.open(format=FORMAT,channels=1,rate=SAMPLEFREQ,input=True,frames_per_buffer=FRAMESIZE)
data = stream.read(NOFFRAMES*FRAMESIZE)
decoded = np.fromstring(data, 'Int16') / ((2**15)/2-1)
stream.stop_stream()
stream.close()
p.terminate()

# Triggerfunktion schneidet bis Schwellenwert alles ab
start = np.argmax(np.abs(decoded) > 0.2)
# Berechnung Endwert der Aufnahme
end = start + 44100 #abtastintervall * 44100 ergibt 1 sec
triggered = decoded[start:end]
#0er anhängen wenn nötig
triggered = np.concatenate((triggered, [0]*(44100 - end - start)))
np.savetxt("rechts5_training.csv", triggered, delimiter=",")

plt.plot(triggered)
plt.title('Aufnahme von "Hallo" Getriggert')
plt.xlabel('Time')
plt.ylabel('Amplitude [Digital Number]')
```

```
#plt.savefig('AufnahmeHalloGetriggert.png')
plt.show()

plt.plot(decoded)
plt.title('Aufnahme von "Hallo" Original')
plt.xlabel('Time')
plt.ylabel('Amplitude [Digital Number]')
#plt.savefig('AufnahmeHalloOriginal.png')
plt.show()
```

Listing 4.2: Skript Versuch 1b

```
import numpy as np
import matplotlib.pyplot as plt

data = np.genfromtxt("hallotriggered.csv", delimiter=",", unpack=True, skip_header=0)

abtastintervall = 0.022957424252209337 / 1000 #sec
signallaenge = 44100

fourier = np.fft.fft(data)

spektrum = np.abs(fourier)

haelfte = int(signallaenge / 2)

print(abtastintervall * signallaenge)

freq = np.zeros(signallaenge)

# Formel um die Anzahl der Schwingungen in die Freuquenz umzurechnen – f = n / (M * delta t)
for n in range(0, signallaenge, 1):
    freq[n] = n/(abtastintervall * signallaenge)

# Darstellung des Amplitudenspektrums
plt.plot(freq[:haelfte], spektrum[:haelfte], "-")
plt.grid()
plt.title('Amplitudenspektrum "Hallo"')
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude [Digital Number]')
plt.savefig('AmplitudenspektrumBreit.png')
plt.show()

```

```
# Darstellung des Amplitudenspektrums
plt.plot(freq[:2500], spektrum[:2500], "-")
plt.title('Reingezoomtes Amplitudenspektrum "Hallo"')
plt.grid()
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude [Digital Number]')
plt.savefig('AmplitudenspektrumSchmal.png')
plt.show()
```

Listing 4.3: Skript Versuch 1c

```
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt

data = np.genfromtxt("rechts1.csv", delimiter=",", unpack=True, skip_header=0)

anzahlAbschnitte = 170
laengeAbschnitt = 512
breiteUeberlappung = int(laengeAbschnitt / 2)

signallaenge = anzahlAbschnitte * breiteUeberlappung

#window = np.zeros(anzahlAbschnitte * breiteUeberlappung)
window = np.ndarray(shape=(anzahlAbschnitte,signallaenge))
gaussianwindow = signal.gaussian(laengeAbschnitt, std=512/4)

for y in range(0, anzahlAbschnitte - 1):
    temp = np.zeros(signallaenge)
    von = int(y * breiteUeberlappung)
    bis = int(von + laengeAbschnitt)
    #print(y)
    temp[0:von] = np.zeros(von)
    temp[von:bis] = data[von:bis] * gaussianwindow
    temp[bis:signallaenge] = np.zeros(signallaenge - bis)

    window[y] = np.abs(np.fft.fft(temp))

# Darstellung des Amplitudenspektrums
plt.plot(gaussianwindow)
plt.grid(True)
plt.title('Gaußsche Fensterfunktion')
plt.xlabel('Samples')
```

```
plt.ylabel('Amplitude')
#plt.savefig('Gauss.png')
plt.show()

meanWindow = np.zeros(signallaenge)

for y in range(0, signallaenge):
    summe = 0
    for x in range(0, anzahlAbschnitte):
        summe = summe + window[x,y]

    meanWindow[y] = summe / anzahlAbschnitte


abtastintervall = 0.022957424252209337 / 1000 #sec
signallaenge = anzahlAbschnitte * breiteUeberlappung
haelfte = int(signallaenge / 2)
freq = np.zeros(signallaenge)

# Formel um die Anzahl der Schwingungen in die Freuquenz umzurechnen - f = n / (M * delta t)
for n in range(0, signallaenge, 1):
    freq[n] = n/(abtastintervall * signallaenge)

# Darstellung des Amplitudenspektrums
plt.plot(freq[:haelfte], meanWindow[:haelfte], "-")
plt.grid()
plt.title('Amplitudenspektrum "Hallo" mit Windowing')
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude [Digital Number]')
#plt.savefig('AmplitudenspektrumBreitWindow.png')
plt.show()

# Darstellung des Amplitudenspektrums
plt.plot(freq[:2500], meanWindow[:2500], "-")
plt.title('Reingezoomtes Amplitudenspektrum "Hallo" mit Windowing')
plt.grid()
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude [Digital Number]')
#plt.savefig('AmplitudenspektrumSchmalWindow.png')
plt.show()
```

Listing 4.4: Skript Versuch 1d

A.1.2 Quellcode Versuch 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

def calcSpectrum(data):
    anzahlAbschnitte = 170
    laengeAbschnitt = 512
    breiteUeberlappung = int(laengeAbschnitt / 2)

    signallaenge = anzahlAbschnitte * breiteUeberlappung

    window = np.ndarray(shape=(anzahlAbschnitte,signallaenge))
    gaussianwindow = signal.gaussian(laengeAbschnitt, std=512/4)

    for y in range(0, anzahlAbschnitte - 1):
        temp = np.zeros(signallaenge)
        von = int(y * breiteUeberlappung)
        bis = int(von + laengeAbschnitt)
        #print(y)
        temp[0:von] = np.zeros(von)
        temp[von:bis] = data[von:bis] * gaussianwindow
        temp[bis:signallaenge] = np.zeros(signallaenge - bis)

        window[y] = np.abs(np.fft.fft(temp))


    meanWindow = np.zeros(signallaenge)

    for y in range(0, signallaenge):
        summe = 0
        for x in range(0, anzahlAbschnitte):
            summe = summe + window[x,y]

        meanWindow[y] = summe / anzahlAbschnitte

    return meanWindow


def main():
    signallaenge = 43520 #damit es genau passt mit den window bereichen
```

```
41
42  data = [np.zeros(signallaenge),np.zeros(signallaenge),np.zeros(signallaenge),np.zeros(signallaenge),np.zeros(signallaenge)]
43
44  typ = "rechts" #hoch, tief, links, rechts
45
46  for n in range(5):
47      pathToFile = typ + str(n + 1) + '.csv'
48      data[n] = np.genfromtxt(pathToFile, delimiter=",", unpack=True, skip_header=0)
49
50  specs = np.zeros((5,signallaenge))
51  for n in range(5):
52      specs[n] = calcSpectrum(data[n])
53
54  mean = np.zeros(signallaenge)
55  for n in range(signallaenge):
56      for i in range(5):
57          mean[n] = mean[n] + (specs[i][n] / 5)
58
59  abtastintervall = 0.022957424252209337 / 1000 #sec
60  haelfte = int(signallaenge / 2)
61  freq = np.zeros(signallaenge)
62  # Formel um die Anzahl der Schwingungen in die Freuquenz umzurechnen -> f = n / (M * delta t)
63  for n in range(0, signallaenge, 1):
64      freq[n] = n/(abtastintervall * signallaenge)
65
66  np.savetxt(typ + "Mean.csv", mean, delimiter=",")
67
68  # Darstellung des Amplitudenspektrums
69  plt.plot(freq[:haelfte], mean[:haelfte], "-")
70  plt.grid()
71  plt.title('Referenzspektrum "' + typ + '"')
72  plt.xlabel('Frequency in Hz')
73  plt.ylabel('Amplitude [Digital Number]')
74  plt.savefig('Referenzspektrum' + typ + '.png')
75  plt.show()
76
77
78 main()
```

Listing 4.5: Skript Versuch 2a

```
1 import numpy as np
2 from scipy import stats
```

```
3
4  hochMean = np.genfromtxt('hochMean.csv', delimiter=",", unpack=True, skip_header=0)
5  tiefMean = np.genfromtxt('tiefMean.csv', delimiter=",", unpack=True, skip_header=0)
6  linksMean = np.genfromtxt('linksMean.csv', delimiter=",", unpack=True, skip_header=0)
7  rechtsMean = np.genfromtxt('rechtsMean.csv', delimiter=",", unpack=True, skip_header=0)
8
9  directory = "person1/"
10
11 commands = ["hoch", "tief", "links", "rechts"]
12 names = []
13 for c in commands:
14     for n in range(1, 6):
15         names.append(c + str(n) + "_training.csv")
16
17 for name in names:
18     spec = np.genfromtxt(directory + name, delimiter=",", unpack=True, skip_header=0)
19     spec = spec[:43520]
20     spec = np.abs(np.fft.fft(spec))
21
22     korr = np.zeros(4)
23
24     korr[0] = stats.pearsonr(spec, hochMean)[0]
25     korr[1] = stats.pearsonr(spec, tiefMean)[0]
26     korr[2] = stats.pearsonr(spec, linksMean)[0]
27     korr[3] = stats.pearsonr(spec, rechtsMean)[0]
28
29     if(np.max(korr) == korr[0]):
30         print("Lösung: " + name + " – Geschätzt: Hoch mit koeff.: " + str(korr[0]))
31     elif(np.max(korr) == korr[1]):
32         print("Lösung: " + name + " – Geschätzt: Tief mit koeff.: " + str(korr[1]))
33     elif(np.max(korr) == korr[2]):
34         print("Lösung: " + name + " – Geschätzt: Links mit koeff.: " + str(korr[2]))
35     elif(np.max(korr) == korr[3]):
36         print("Lösung: " + name + " – Geschätzt: Rechts mit koeff.: " + str(korr[3]))
```

Listing 4.6: Skript Versuch 2c+d