Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Signale, Systeme und Sensoren

AUFBAU EINES EINFACHEN SPRACHERKENNER

Kiattipoom Pensuwan, Thanh Son Dang

Konstanz, 19. Dezember 2018

Zusammenfassung

Thema:	AUFBAU EINES EINFACHEN SPRACHERKENNER	
Autoren:	Kiattipoom Pensuwan	ki851pen@htwg-konstanz.de
	Thanh Son Dang	th851dan@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz	mfranz@htwg-konstanz.de
	Jürgen Keppler	juergen.keppler@htwg-konstanz.de
	Simon Christofzik	si241chr@htwg-konstanz.de

In diesem Versuch wird ein einfacher Spracherkenner aufgebaut, welcher nur vier einfache Befehle "Hoch", "Tief", "Rechts" und "Links" erkennen kann. Damit diese Befehle erkannt werden, müssen die Referenzspektren erstellt werden und mit den aufgenommenen akustischen Signalen verglichen.

Um die Referenzspektren zu bekommen, muss jeder Befehl mehrmals aufgenommen, mit der Windowing-Methode zerlegen, die lokale Fouriertransformation in jedem Fenster durchführen und daraus den Mittelwert abbilden. Mit diesen Referenzspektren und der Mustererkennung durch Korrelation(Korrelationskoeffizienten nach Bravais-Pearson) können das gesprochen Wort erkannt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1

Fourieranalyse lang andauernder Signale

1.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Ein beliebige Spracheingabe wird über die Soundkarte mithilfe des Objektes audiorecorder aufgenommen und als akustisches Signal gespeichert. Das originale Signal wird grafisch dargestellt.

Dann wird dieses Signal so, dass alle Sample nahe an 0 (kleiner als 10 Prozent von Maximalen Wert) liegen, abgeschnitten(Triggerfunktion), damit alle Signale beginnen nur wenn die eigentliche Spracheingabe angefangen hat. Das neue abgeschnittene Signal sollte genau 1 s lang sein. Zur Überprüfung der Korrekheit der Triggerfunktion werden die beide grafisch dargestellten Signale verglichen.

Mithilfe der Funktion numpy.fft.fft() kann man die Fouriertransformierte des abgeschnitten Signals berechnen. Daraus wird das Amplitudenspektrum bestimmt und grafisch dargestellt.

Das Signal wird mit der Methode des Windowing in Abschnitte mit der Länge von 512 Samples zerlegen, die sich jeweils zur Hälfte überlappen. In jedem Fenster werden die Samples mit der Gaußschen Fensterfunktion multipliziert, die die Fensterbreite 4 Standardabweichungen hat. Dadurch werden plötzliche Sprünge vermieden.

In jedem Fenster wird eine lokale Fouriertransformation durchgeführt und daraus den Mittelwert bilden und wieder grafisch darstellen. Die Korrektheit wird durch Vergleich mit dem ohne Windowing Spektrum überprüft.

1.2 Messwerte

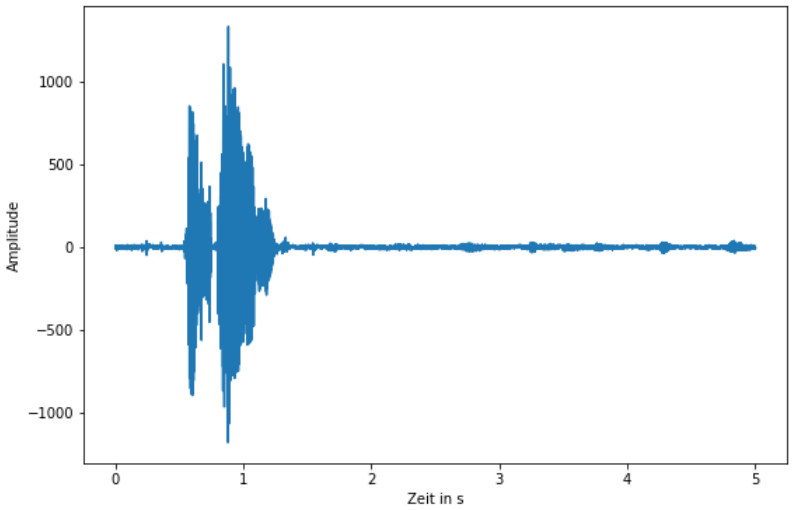

Abbildung 1.1: Das komplette aufgenommene Signal von den Worten "was cooles"

1.3 Auswertung

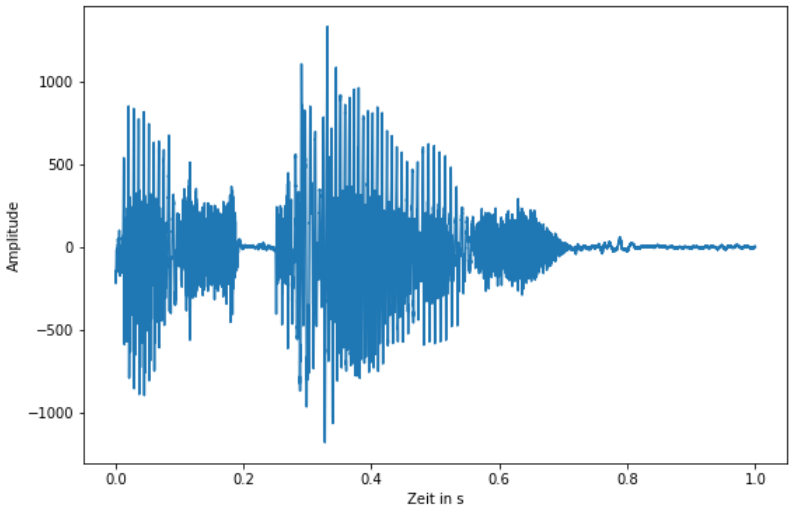

Abbildung 1.2: Das abgeschnittene Signal von den Worten "was cooles"

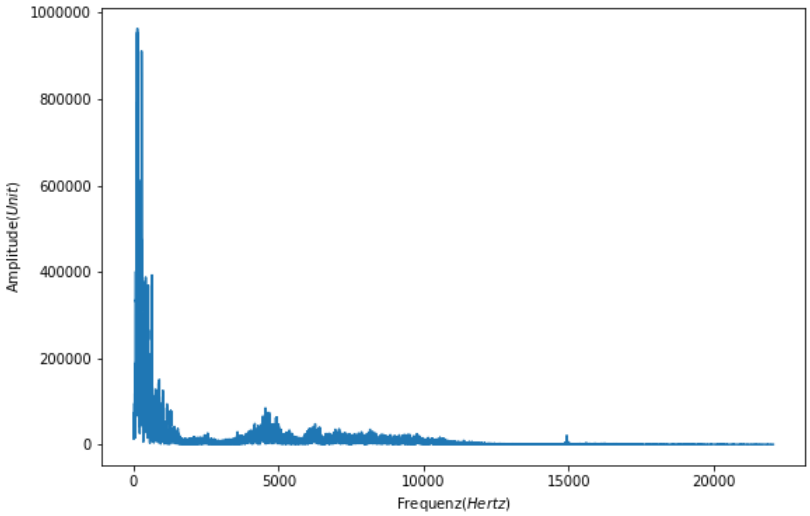

Abbildung 1.3: Das Amplitudenspektrum von dem abgeschnittenen Signal

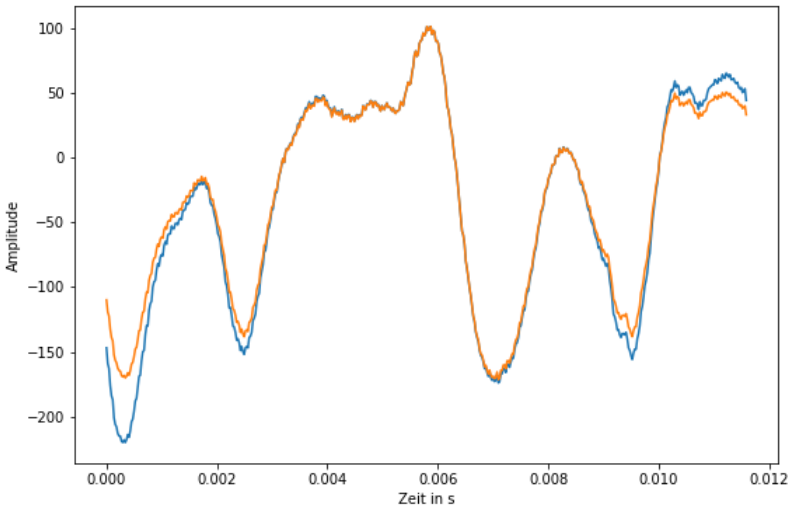

Abbildung 1.4: Beispiel von einem Fenster(Blau:ohne Fensterfunktion, Orange:mit Fensterfunktion)

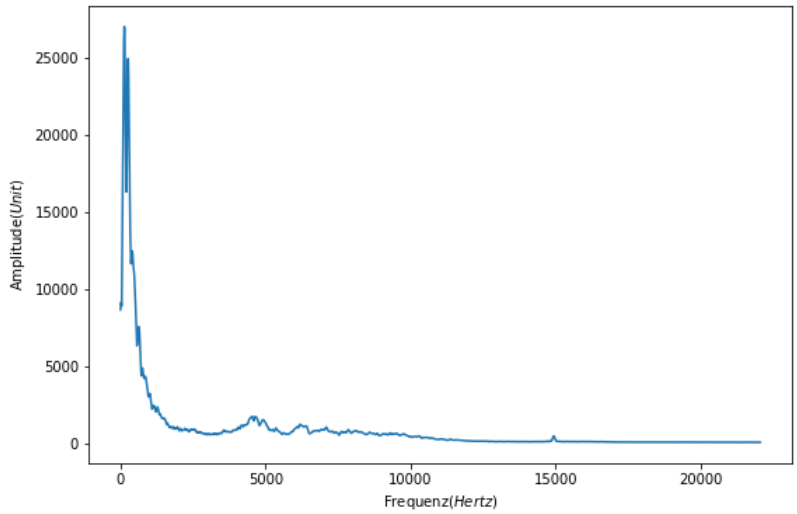

Abbildung 1.5: Amplitudenspektrum durch Mittelung aller lokalen Fouriertransformierten

Zusätzlich haben wir auch jede lokalen Fouriertransformierte auf einem Graph dargestellt:

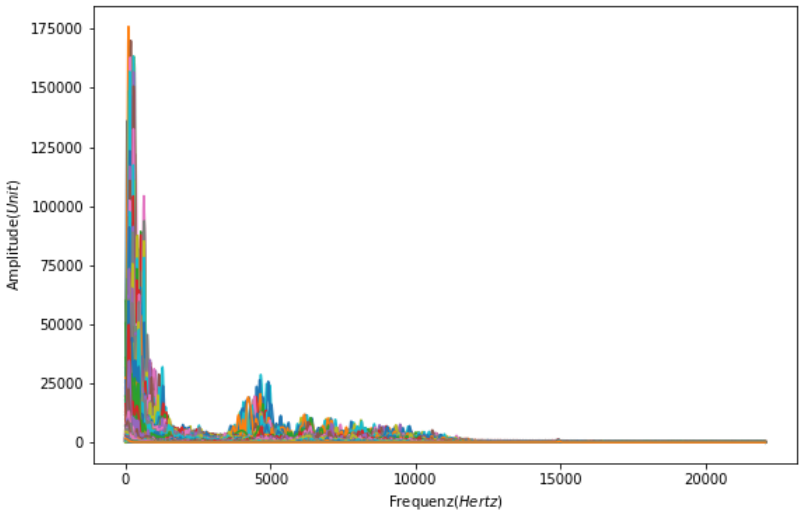

Abbildung 1.6: Amplitudespektrum aller lokalen Fouriertransformierten

Den Code davon bitte den Anhang A.1.1 und A.1.2 sehen.

1.4 Interpretation

Durch die Amplitudenspektren(Abbildungen 1.3, 1.5 und 1.6) kann festgestellt werden, dass die die ähnlichen Verhältnissen vorliegen aber die Amplituden variieren. Das heißt, jede Frequenz, die im Spektrum des abgeschnitten Signal aufgetreten ist, ist auch im Spektrum aller Fenster aufgetreten, aber diese Frequenzen kommen nicht so häufig wie im originalen Signal vor, weil außerhalb der Fenster sind die Amplituden von Signal 0(manche Frequenzen kommen gar nicht vor).

2

Spracherkennung

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Die vier Befehle "Hoch", "Tief", "Links" und "Rechts"werden jeweils 5 Mal von gleichem Sprecher aufgenommen und mit der Windowing-Methode von vorherigem Versuch deren Spektren berechnen. Durch die Mittelung dieser Spektren ergibt sich die Referenzspektren von den 4 Befehlen. Die Referenzspektren werden für den Spracherkenner benötigt, indem wir die Berechnung der Korrelationskoeffizienten nach Bravais-Pearson zum Vergleich zweier Spektren(Referenzspektrum und Eingabespektrum) durchführen. Beim Vergleich identischer Spektren sollte die Korrelation 1 sein, bei verschiedenen Spektren nahe an 0.

Beim Spracherkenner werden Korrelationskoeffizient von dem Eingabespektrum mit jedem Referenzspektrum berechnet. Dann werden die Korrelationskoeffizienten miteinander verglichen. Das Wort hat die Korrelationskoeffizient, die am meisten in der Nähe von 1 liegt, wird ausgewählt.

Zum Testen des Spracherkenners werden die oben genannten Befehle jeweils noch 5 Mal von 2 Sprechern aufgenommen. Die Detektions- und die Fehlerrate wird notiert.

2.2 Messwerte

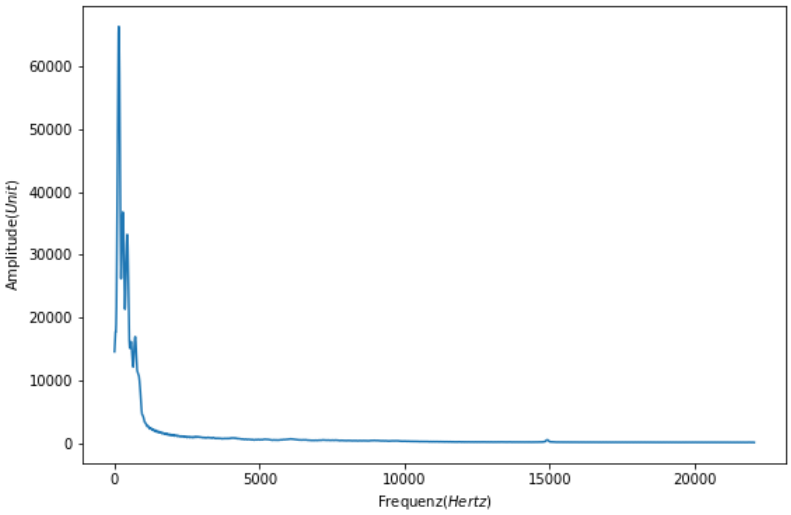

Abbildung 2.1: Referenzspektrum von dem Befehl "Hoch"

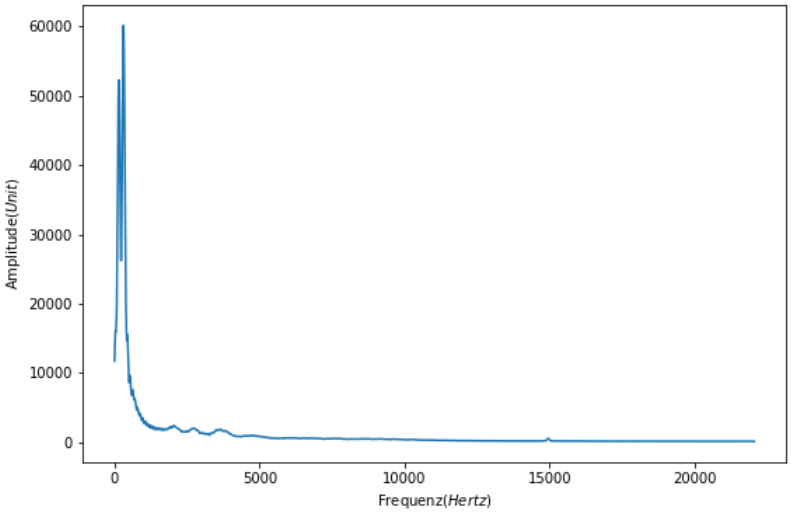

Abbildung 2.2: Referenzspektrum von dem Befehl "Tief"

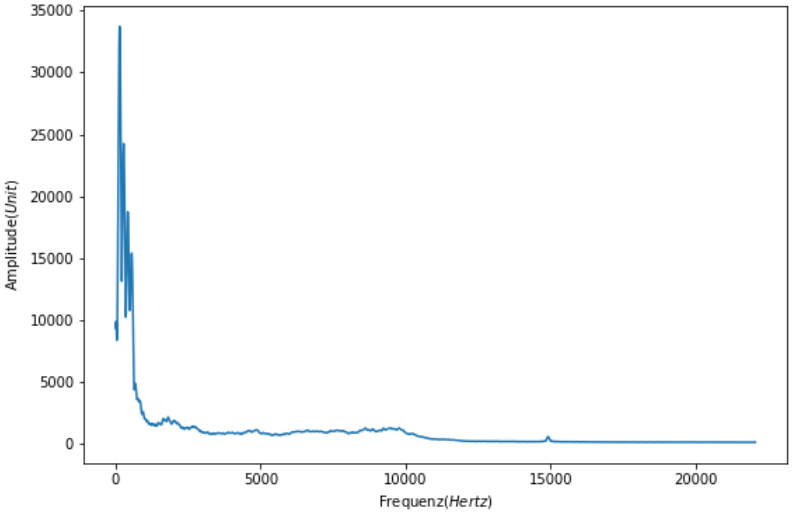

Abbildung 2.3: Referenzspektrum von dem Befehl "Rechts"

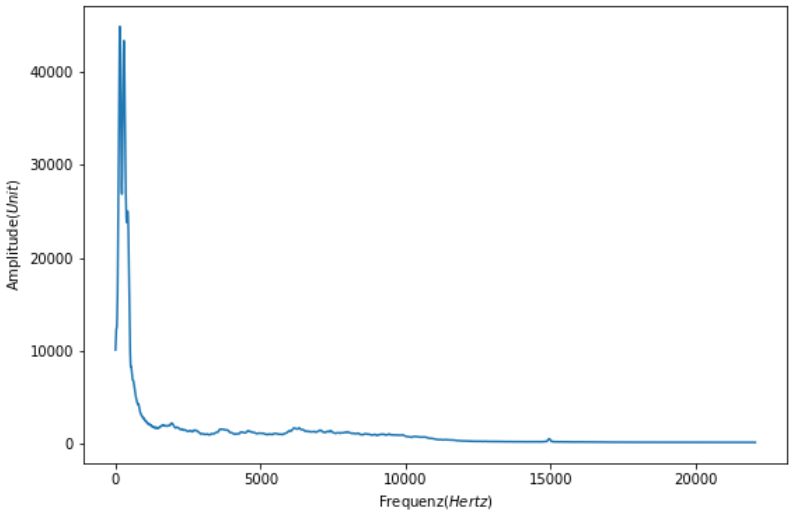

Abbildung 2.4: Referenzspektrum von dem Befehl "Links"

2.3 Auswertung

Korrelationskoeffizient nach Bravais-Pearson:

$$r_{fg} = \frac{\sigma_{fg}}{\sigma_f \cdot \sigma_g}$$

indem σ_f und σ_g jeweils Standardabweichungen von den entsprechenden Signalen f und g sind, die nach der Formel berechnet:

$$\sigma_f^2 = \frac{1}{n-1} \sum_{k=1}^{n} (f_k - \mu_f)^2$$

Die Kovarianz σ_{fg} ist für diskrete Signale definiert als:

$$\sigma_{fg} = \frac{1}{n} \sum_{k=1}^{n} (f_k - \mu_f) \cdot (g_k - \mu_g)$$

μ_f ist *der Mittelwert:*

$$\mu_f = \frac{1}{n} \sum_{k=1}^{n} f_k$$

Befehl	Sprecher 1	Sprecher 2
Hoch	100%	100%
Tief	100%	80%
Links	100%	100%
Rechts	100%	80%

Tabelle 2.1: Detektionsrate

Den Code davon bitte den Anhang A.1.3 und A.1.4 sehen.

2.4 Interpretation

Die Detektion besagt, dass beim Sprecher 1 100% das Wort korrekt erkannt wurde und beim Sprecher 2 ist abweichend bei der Erkennung der Befehle "Tief" und "Rechts". Das ist total verständlich, weil der Sprecher 1 ist die Person, von der die Referenzspektren hergestellt wurden. Bei dem anderen Sprecher wurde die Befehlen manchmal falsch erkannt. Die Erklärung dafür ist aus den anatomischen Gründe, dass jede Person eigene Stimme und eigene Sprechweise(Akzent) hat. Die Fehlerrate ist nicht so hoch, da die Datenbank der Referenzspektrum noch ziemlich klein ist (von nur 4 Befehlen).

Anhang

A.1 Quellcode

A.1.1 Quellcode Signal Aufnehmen, Triggerfunktion und Fourriertransformation

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

FORMAT = pyaudio.paInt16
SAMPLEFREQ = 44100
FRAMESIZE = 1024
NOFFRAMES = 220
INPUT_BLOCK_TIME = 0.05
INPUT_FRAMES_PER_BLOCK = int(SAMPLEFREQ*INPUT_BLOCK_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 = np.fromstring(data, 'Int16')
stream.stop_stream()
stream.close()
p.terminate()
print('done')

#######Signal speichern und darstellen########
string = 'hoch1'
np.save(string,decoded)
sec = len(decoded) / SAMPLEFREQ
plt.xlabel('Zeit in s')
plt.ylabel('Amplitude')
```

```
Zeit = []
for i in range (len(decoded)):
    Zeit.append(sec/len(decoded) * i)
plt.plot(Zeit,decoded)
plt.show()

####Signal mit Triggerfunktion abschneiden####
trigger = 0.1 * np.max(decoded)
j = 0
for i in decoded:
    j = j + 1
    if np.abs(i) > trigger:
        decoded = decoded[j:j+SAMPLEFREQ]
        break
np.save(string+'abgeschnitten',decoded)
plt.xlabel('Zeit in s')
plt.ylabel('Amplitude')
Zeit = []
for i in range (len(decoded)):
    Zeit.append(1/len(decoded) * i)
plt.plot(Zeit,decoded)
plt.show()

############Fourriertranformation###############
spek = abs(np.fft.fft(decoded))
plt.plot(spek)
plt.savefig('Amplitudenspektrum.png')
```

A.1.2 Quellcode Windowing und Spektrum Versuch 1

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

def fft(data,name):
    plt.figure(figsize=(9,6))
    Y = abs(np.fft.fft(data))
    Y = Y[range(int(len(Y)/2))]
    x = np.linspace(0,22050,22050,endpoint = True)
    plt.xlabel('Frequenz($Hertz$)')
    plt.ylabel('Amplitude($Unit$)')
    plt.plot(x,abs(Y))
```

```
    plt.savefig(str(name)+".png")
    plt.show()

def winspek(data,name):
    plt.figure(figsize=(9,6))
    plt.xlabel('Frequenz($Hertz$)')
    plt.ylabel('Amplitude($Unit$)')
    st = np.std(data)
    from scipy import signal
    gfen = signal.gaussian(512, std = st * 4)
    g = np.zeros(len(data))
    fft = np.zeros(len(data))
    x = np.linspace(0,22050,22050,endpoint = True)
    for i in range(0,len(data),256):
        if (i > len(data) - 512):
            gfen = signal.gaussian(len(data)-i, std = st * 4)
            g[i:] = data[i:] * gfen
            fft += abs(np.fft.fft(g))
            a = np.fft.fft(g)
            a = a[range(int(len(a)/2))]
            plt.plot(x,abs(a))
            break
        g[i:i+512] = data[i:i+512] * gfen
        fft += abs(np.fft.fft(g))
        a = np.fft.fft(g)
        a = a[range(int(len(a)/2))]
        plt.plot(x,abs(a))
        g = np.zeros(len(data))
    plt.savefig("spektrum_gesamter_windows"+str(name)+".png")
    plt.show()
    fft /= 171 #durch die Anzahl der Windows teilen
    fft = fft[range(int(len(fft)/2))]
    plt.figure(figsize=(9,6))
    plt.xlabel('Frequenz($Hertz$)')
    plt.ylabel('Amplitude($Unit$)')
    plt.plot(x,fft)
    plt.savefig("spektrum_mittel_windows"+str(name)+".png")
    return fft

data = np.load('was_cooles_abgeschnitten.npy')
plt.figure(figsize=(9,6))
plt.xlabel('Zeit in s')
```

```
plt.ylabel('Amplitude')
Zeit = []
for i in range (len(data)):
    Zeit.append(1/len(data) * i)
plt.plot(Zeit,data)
plt.savefig("v1_abgeschnitten.png")
plt.show()
fft(data,"Amplitudenspektrum")

winspek(data,"v1")
```

A.1.3 Quellcode Windowing und Referenzspektrum Versuch 2

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

def winspek(data):
    st = np.std(data)
    gfen = signal.gaussian(512, std = st * 4)
    g = np.zeros(len(data))
    fft = np.zeros(len(data))
    for i in range(0,len(data),256):
        if (i > len(data) - 512):
            break
        g[i:i+512] = data[i:i+512] * gfen
        fft += abs(np.fft.fft(g))
        a = np.fft.fft(g)
        a = a[range(int(len(a)/2))]
        g = np.zeros(len(data))

    fft /= 171 #durch die Anzahl der Windows teilen
    fft = fft[range(int(len(fft)/2))]
    return fft

data = np.load('hoch1abgeschnitten.npy')
mit =  np.zeros(int(len(data)/2))
a = ['hoch', 'tief', 'rechts', 'links']
for j in a:
    mit = 0
    for i in range(1,6):
```

```
        data = np.load(j + str(i) + 'abgeschnitten.npy')
        np.save('spek' + j + str(i),winspek(data))
        mit+=winspek(data)
    mit = mit/5
    np.save('ref' + j,mit)
    plt.figure(figsize=(9,6))
    plt.xlabel('Frequenz($Hertz$)')
    plt.ylabel('Amplitude($Unit$)')
    plt.plot(mit)
    plt.savefig("spektrum_mittel_windows" + j + ".png")
    plt.show()
```

A.1.4 Quellcode Korrelation und Spracherkenner Versuch 2

```
import numpy as np

refhoch = np.load('experiment 2a/refhoch.npy')
reftief = np.load('experiment 2a/reftief.npy')
refrechts = np.load('experiment 2a/refrechts.npy')
reflinks = np.load('experiment 2a/reflinks.npy')
def spracherkenner(name,person):
    for i in range(1,6):
        test = np.load('Spek/spek'+str(name)+str(i)+'t'+str(person)+'.npy')
        korrcoefh = np.corrcoef(refhoch, y = test)
        korrcoeft = np.corrcoef(reftief, y = test)
        korrcoefr = np.corrcoef(refrechts, y = test)
        korrcoefl = np.corrcoef(reflinks, y = test)
        a = np.max([np.mean(korrcoefh), np.mean(korrcoeft), np.mean(korrcoefr), np.mean(korrcoefl)])
        if a == np.mean(korrcoefh):
            print('hoch')
        elif a == np.mean(korrcoeft):
            print('tief')
        elif a == np.mean(korrcoefr):
            print('rechts')
        elif a == np.mean(korrcoefl):
            print('links')

spracherkenner('links','p')
print("----")
spracherkenner('links','s')
```