Hochschule Konstanz
Fakultät Informatik

Signale, Systeme und Sensoren

Aufbau eines einfachen Spracherkenners

J. Graff, D. Schellinger, T. Mack

Konstanz, 14. Juni 2021

Zusammenfassung (Abstract)

Thema:	Aufbau eines einfachen Spracherkenners	
Autoren:	J. Graff	johannes.graff@htwg-konstanz.de
	D. Schellinger	daniel.schellinger@htwg-konstanz.de
	T. Mack	tobias.mack@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

1

Einleitung

In diesem Versuch wird ein einfacher Spracherkenner aufgebaut, der z.B. zur Steuerung eines Staplers in einem Hochregallager dienen könnte. Es reichen hierzu die Erkennung der vier einfachen Befehle 'Hoch', 'Tief', 'Links' und 'Rechts'. Der Aufbau des Spracherkenners folgt dem in der Vorlesung beschriebenen Prinzip des Prototyp-Klassifikators. Die zugehörigen Spektren werden mit der Windowing-Methode berechnet. Die Vorüberlegungen, Schaltungen, Berechnungen und Ergebnisse sind wie immer zu protokollieren.

2

Versuch 1

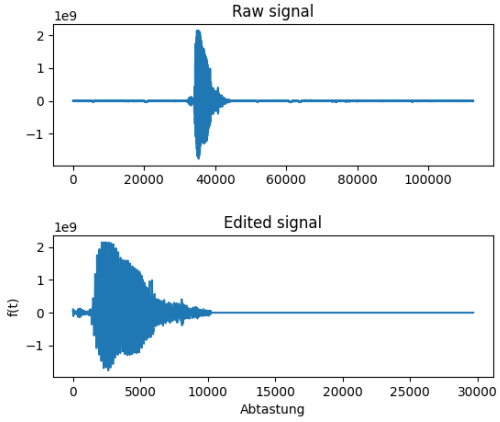
2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Mit Hilfe der Bibliothek Pyaudio wird wie im Laborversuch 3, Audiosignale aufgenommen und als Datei abgespeichert. Die Aufgenommenen Signale werden so editiert, dass sie bei Sekunde 0 beginnen und Rauschen nach dem gesprochenen Wort entfernt wird. Zusätzlich sollte das Signal eine zeitliche Dauer von einer Sekunde aufweisen. Das Abschneiden des Anfangs in dem kein Laut entstanden ist, durch eine Schleife ermöglicht, in der das Signal nach einem bestimmten Amplitudenschwellwert untersucht und nach Überschreitung des Schwellenwertes den Teil mit einem gewissen offset hinter sich löscht.Um das Rauchen am Ende zu unterdrücken, werden die Daten mit der Funktion np.flip gespiegelt, und wie zuvor vorgegangen. Statt das Signal zu löschen, werden die Daten bis einen bestimmten Schwellwert gleich 0 gesetzt.Der letzte Schritt der Bearbeitung, ist das ausschneiden des Signales um eine genaue Zeitdauer von einer Sekunde zu erhalten.Nach der Bearbeitung des Signals wird wie in Laborversuch 3, das Amplitudenspektrum des Signals berechnet und in einem Plot dargestellt.

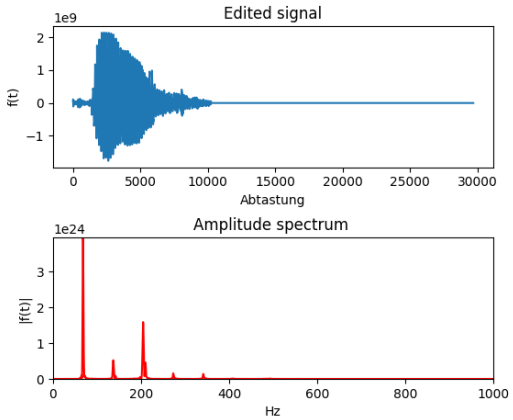
Das Windowing wurde mit einer Länge von 512 Samples durchgeführt, wobei sich jeweils 256 Samples überschneiden. Zur Gewichtung wurde in jedem Fenster eine Gaußsche Fensterfunktion benutzt, um plötzliche Sprünge im Spektrum zu vermeiden. Dabei wurde die numpy Funktion $numpy.kaiser(512,\ 4)$ verwendet. In jedem Fenster wird also eine lokale Fouriertransformation durchgeführt und dann über alle Fenster gemittelt, um das Amplitudenspektrum zu erhalten.

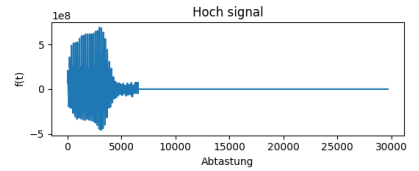
2.2 Messwerte

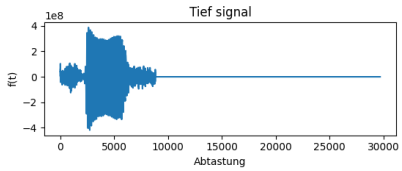
Um den Vorgang der Bearbeitung zu überprüfen, wird das Signal in einem Plot dargestellt und verglichen.

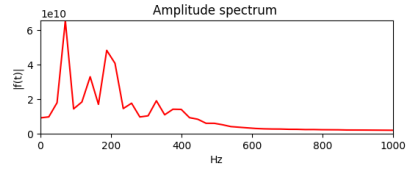


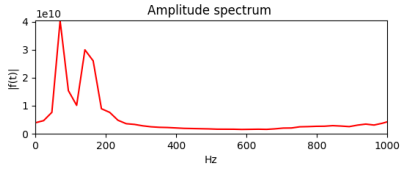
Durch das relativ kurze Signal sind die verschiedenen Frequenzen leicht unklar dargestellt.

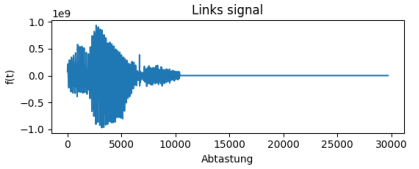


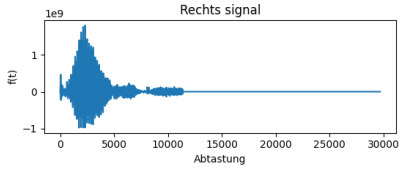


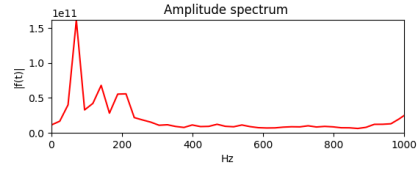


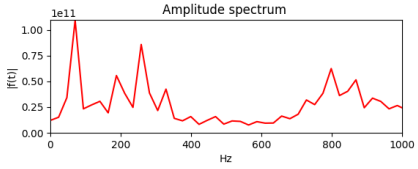












2.3 Auswertung

Im Vergleich zu der einzelnen Aufnahme sind die Referenzspektren mit Windowing etwas verrauschter bzw. ungenauer, da die 5 verschiedenen Aufnahmen eines Wortes leicht unterschiedlich hohe Frequenzen enthalten und ein gewisses Grundrauschen bei der Aufnahmequalität ebenfalls variiert. Durch das Windowing in Verbindung mit der Mittlung der 5 Aufnahmen, wird aber ein Wort genauer erkannt.

3

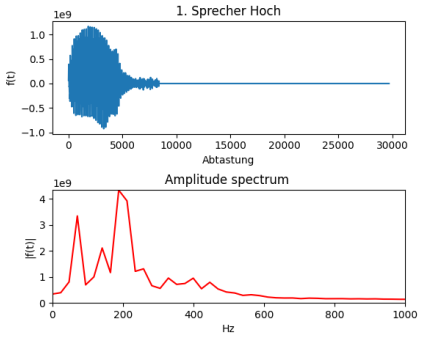
Versuch 2

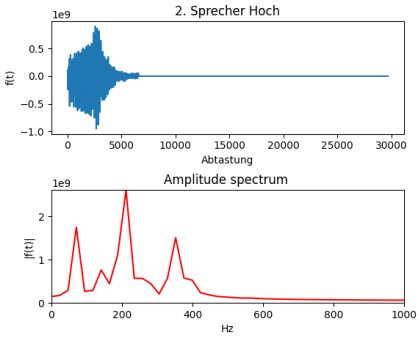
Die Referenzspektren für die Wörter 'Hoch', 'Tief', 'Links' und 'Rechts' wurden jeweils 5 mal von Daniel eingesprochen. Anschließend wurden die Beispiele jeweils gemittelt, um ein Referenzspektrum zu erhalten, das die durchschnittlichen Frequenzen der Wörter erhält und somit für den Spracherkenner verwendet werden kann. Zusätzlich wurde ein Testdatensatz aus wiederum 5 Beispielen zu jedem Wort von Daniel erstellt. Dazu wurde ein weiterer Testdatensatz erstellt, der sich wiederum aus 5 Beispielen zu jedem Wort, gesprochen von Tobias und Johannes, zusammensetzt.

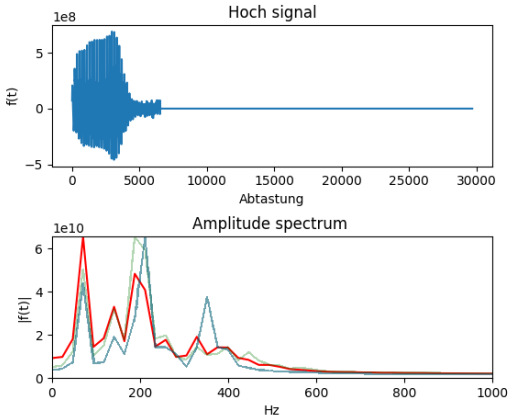
3.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Um die Audioaufnahme nun mit dem Referenzspektrum zu vergleichen, wird ein Korrelationskoeffizient benötigt, der die Genauigkeit angibt und somit einem Wort zugeordnet werden kann. Dazu wurde die Funktion scipy.stats.pearsonr(Referenzspektrum, Spektrum Audioaufnahme) verwendet, um den Bravais-Pearson-Koeffizient zu bestimmen. Anschließend werden die Koeffizienten für die 4 Referenzspektren verglichen, wobei der größte Koeffizientenwert die höchste Übereinstimmung mit einem der 4 Worte / Referenzspektren anzeigt.

3.2 Messwerte







3.3 Auswertung

Der einfache Spracherkenner hat beim Testdatensatz 1 eine Detektionsrate von 90 Prozent, lediglich bei ’Tief’ traten zwei Fehlerkennungen auf. Beim Testdatensatz 2, bestehend aus Beispielen von anderen Sprechern, lag die Detektionsrate bei 85 Prozent, die Fehler traten wiederum bei dem Wort Tief auf, das vom Spracherkenner mit ’Links’ verwechselt wurde. Dies liegt höchstwahrscheinlich an dem dominanten Vokal ’I’, der in beiden Wörtern vorkommt. Es wäre hilfreich, das Referenzspektrum aus mehr als fünf Beispielen zu erstellen, um die Abgrenzung der Wörter ’Links’ und ’Tief’ deutlicher darzustellen. Zudem ist es denkbar, dass eine spezifischere Windowing-Methode sowie kleinere Windows zu einer besseren Spracherkennung führen würden.

Anhang

A.1 Quellcode

A.1.1 Quellcode Versuch 1

```
# Signal editing
#   Trigger cut at 0
print(csvData)
c = 0
for i in csvData:
    c = c + 1
    if i > 100000000:
        csvData2 = np.genfromtxt("messungandere4.txt", dtype=float, encoding = None,skip_header= c - 5)
        break

#   missing samples filed with zeros
c = csvData2.size
c = 0
for i in np.flip(csvData2):
    c = c - 1
    if (i > 50000000):
        break
    csvData2[c] = 0

#   remaining runtime 1000ms
np.flip(csvData2)
signalLength = 29696
csvData2 = csvData2[0:signalLength]
```

```
# Amplitude spectrum
#   calculating Amplitude spectrum (Prog.Code L3_21)
csvData2 = abs(np.fft.fft(csvData2))
power_decoded = np.square(csvData2)
frequency = np.linspace(0, SAMPLEFREQ/2, len(power_decoded))

#   plot result
fig, (ax1, ax2) = plt.subplots(2)
ax1.set(xlabel="Abtastung", ylabel="f(t)")
ax2.set(xlabel="Hz", ylabel="|f(t)|")
ax1.plot(Links1)
ax2.plot(frequency/2, fftDataLL, 'r')
plt.axis([0, 1000, 0, max(fftDataLL)])
ax1.title.set_text('Links Raw signal')
ax2.title.set_text('Amplitude spectrum ')
plt.subplots_adjust(left=0.15,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.5)

fig.show()
```

A.1.2 Quellcode Versuch 2

```
#   Wighting of Sections with Gaussian Window function
#   window width = 4 * Std
window = np.kaiser(512, 4)
print("window Complete")

while i < (signalLength/256)-1:

    #   transforming the Signal into 512ms Sections
    #   with 50% overlap
    #   local fft in each window
    fftData[0:512] = (abs(np.fft.fft(csvData3[(i*256):(512 + (i * 256))]_)) + fftData[0:512])
    fftDataTT1[0:512] = abs(np.fft.fft(TiefTest1[(i * 256):(512 + (i * 256))])) + fftDataTT1[0:512]
```

```
#   calculate Correlation coefficients (Bravis-Pearson)
KoeffT = sp.stats.pearsonr(fftDataT,fftDataTT1)
KoeffH = sp.stats.pearsonr(fftDataH,fftDataTT1)
KoeffL = sp.stats.pearsonr(fftDataL,fftDataTT1)
KoeffR = sp.stats.pearsonr(fftDataR,fftDataTT1)

maxkoef= max(KoeffT[0], KoeffH[0], KoeffL[0], KoeffR[0])*100
print(max(KoeffT[0], KoeffH[0], KoeffL[0], KoeffR[0])*100)

#   comparing reference spectrum
if (KoeffT[0] * 100 == maxkoef):
    print("Tief mit ", maxkoef, "%")

if (KoeffH[0] * 100 == maxkoef):
    print("Hoch mit ", maxkoef, "%")

if (KoeffR[0] * 100 == maxkoef):
    print("Rechts mit ", maxkoef, "%")

if (KoeffL[0] * 100 == maxkoef):
    print("Links mit ", maxkoef, "%")
```