Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Signale, Systeme und Sensoren

Aufbau eines einfachen Spracherkenners

Tobias Schoch, Luca Strattmann

Konstanz, 25. Mai 2019

Zusammenfassung (Abstract)

Thema:	Aufbau eines einfachen Spracherkenners	
Autoren:	Tobias Schoch	tobias.schoch@htwg-konstanz.de
	Luca Strattmann	luca.strattmann@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz	mfranz@htwg-konstanz.de
	Jürgen Keppler	juergen.keppler@htwg-konstanz.de
	Christoph Kaiser	ch241kai@htwg-konstanz.de

Zusammenfassung etwa 100 Worte.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

1

Einleitung

[?] [?]

2

Versuch 1

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Fragestellung:
In dem ersten Teil des Versuchs 'Aufbau eines einfachen Spracherkenners' werden wir einen Spracherkenner bauen. Zudem erweitern wir den Spracherkenner mit einer Triggerfunktion. Mit der Aufnahme bestimmen wir dessen Amplitudenspektrum. Im Anschluss werden die Methode des Windowing anwenden.
So werden wir eine Testaufnahme machen um diese anschließend auszuwerten.
Auf die entstandenen Numpy-Dateien wenden wir Windowing und das Amplitudenspektrum an.

Messprinzip:
Im ersten Versuch starten wir mit einem Pythonskript. Das Mikrofon wird mit einem Klinkenstecker direkt an die Soundkarte des Computers verbunden. Auf den Computern im Labor, ist das Paket PyAudio bereits in den IDE's integriert.
Mit dem geschriebenen Pythonskript können wir nun akustische Signale aufnehmen. Über das Objekt audiorecorder haben wir Zugriff auf die Aufnahmefunktion der Soundkarte.
Das Signal speichern wir anschließend mittels numpy.save(). Im Anschluss sollen wir eine beliebige Spracheingabe aufnehmen und diese in einem Diagramm darstellen.

Im Anschluss sollen wir das Aufnahmeprogramm um eine Triggerfunktion erweitern, welche die Aufnahme erst ab einem gewissen Lautstärkepegel starten lässt.
So können wir sicher stellen, dass alle Aufnahmen den selben Startpunkt besitzen.

Das Signal soll eine Dauer von einer Sekunde haben und die fehlenden Samples mit Nullen augefüllt werden.

Mit dem Code Aus dem dritten Versuch können wir mit der Aufnahme das Amplitudenspektrum bestimmen. Dies stellen wir grapisch dar. Danach implementieren die Methode des Windowing.
Diese werden wir jeweils in einer Länge von 512 Samples darstellen. Die einzelnen Windows werden wir mit der Gaußschen Fensterfunktion multiplizieren die eine Fensterbreite von der Standartabweichung 4 hat.
Den ersten Versuch werden wir mit dem Amplitudenspektrum erneut überprüfen und so das Spektrum aus der letzten Aufgabe auf Korrektheit überprüfen.

Aufbau:
Das Mikrofon wird durch einen Klinkenstecker direkt an die Soundkarte des Computers verbunden. Durch ein Pythonskript können wir nun Sprachaufnahmen machen.

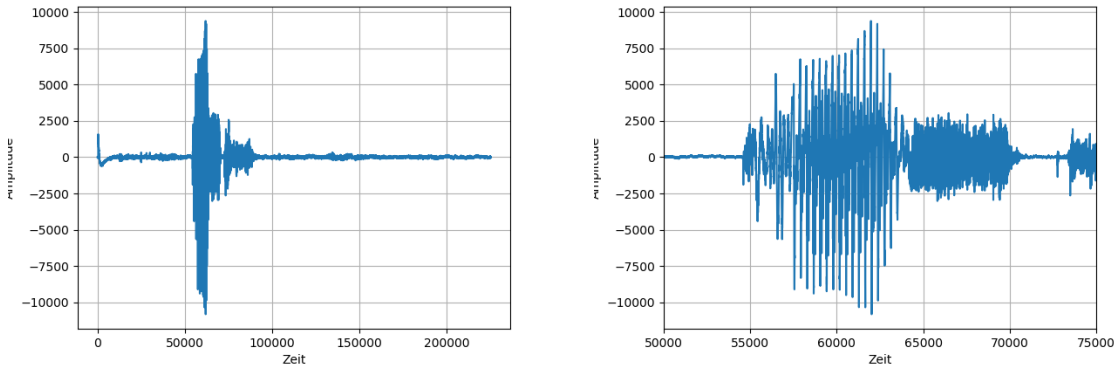
Abbildung 2.1: Aufnahme mit Triggerfunktion des Wortes rechts

Messmittel:

- Ein Mikrofon
- Ein Computer mit einer Python IDE

2.2 Messwerte

Mit einem Mikrofon das an den Computer angeschlossen ist, sehen wir die mit dem Pythonskript aufgenommene Sprachaufnahme des Wortes 'Test', welche mit Python visualisiert wurde. In der linken Darstellung ist die gesamte Dauer des empfangenen Signals. Im rechten Bild ist eine verkürzte Darstellung der Sprachaufnahme um das Signal besser zu erkennen.



(a) Sprachaufnahme des Wortes Test

(b) Dieselbe Sprachaufnahme mit kurzer Zeitachse

Abbildung 2.2: Die aufgenommene Sprachaufnahme visualisiert mit Python

Im folgenden sieht man die Sprachaufnahme des Wortes 'Rechts' mit der Triggerfunktion.

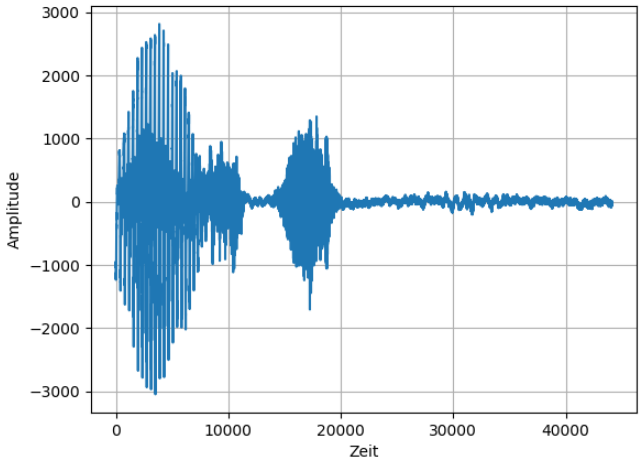
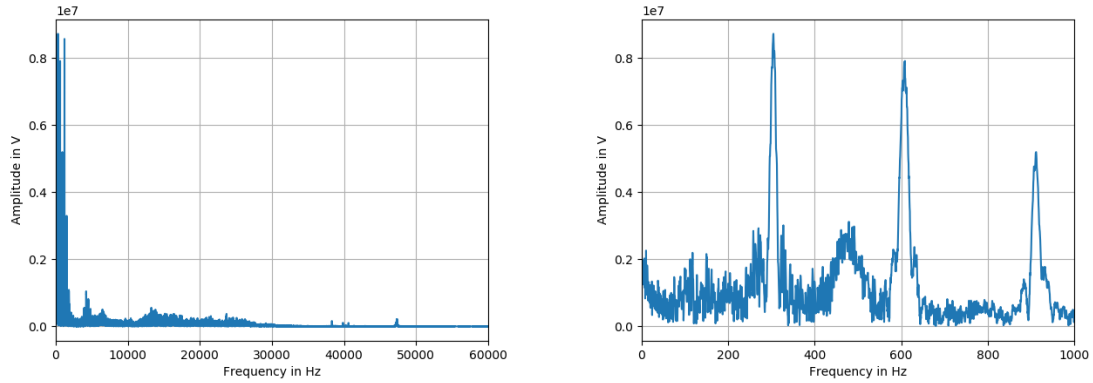


Abbildung 2.3: Aufnahme mit Triggerfunktion des Wortes rechts

2.3 Auswertung



(a) Amplitudenspektrum der Sprachaufnahme (b) Amplitudenspektrum mit geringerer Frequenz

Abbildung 2.4: Das Amplitudenspektrum mit der dazugehörigen Frequenz

Im folgenden sieht man die Sprachaufnahme des Wortes ’Rechts’ mit der Triggerfunktion.

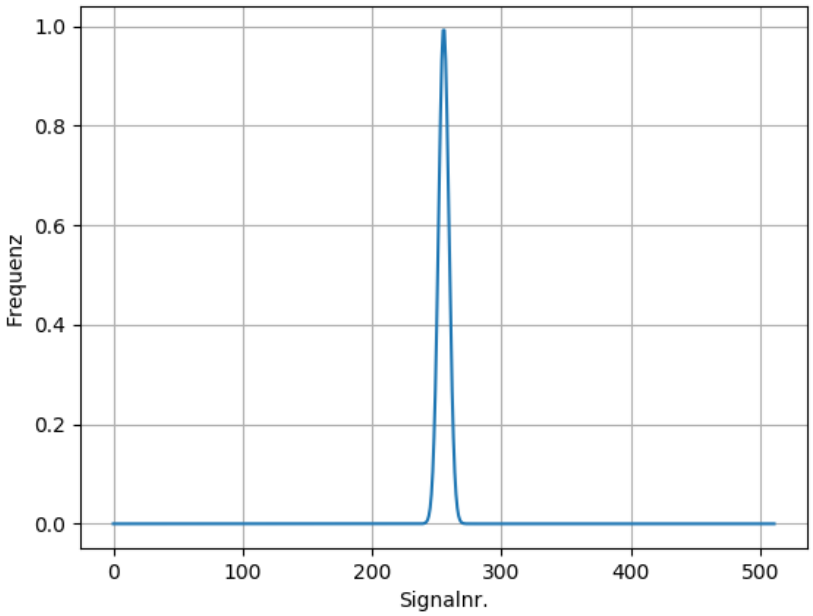


Abbildung 2.5: Aufnahme mit Triggerfunktion des Wortes rechts

Im folgenden sieht man die Sprachaufnahme des Wortes ’Rechts’ mit der Triggerfunktion.

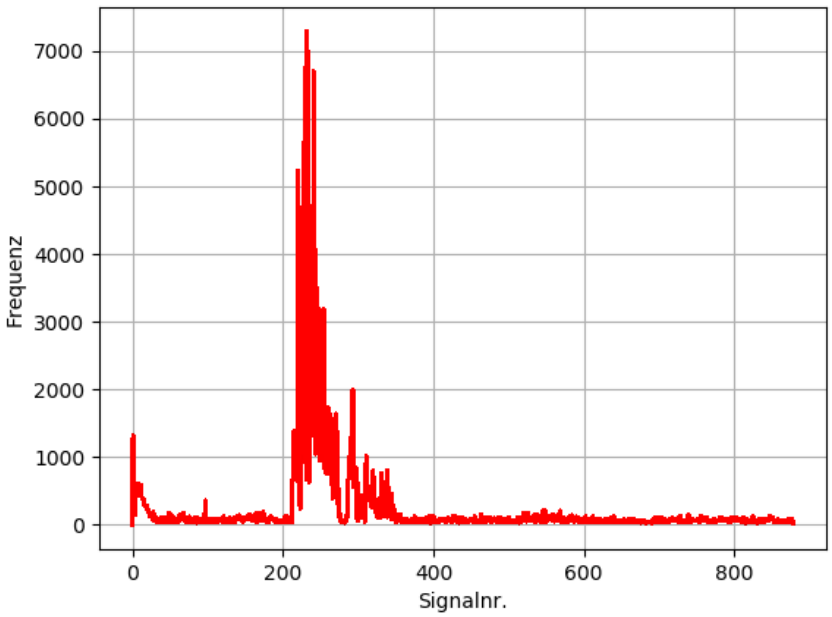


Abbildung 2.6: Aufnahme mit Triggerfunktion des Wortes rechts

Im folgenden sieht man die Sprachaufnahme des Wortes ’Rechts’ mit der Triggerfunktion.

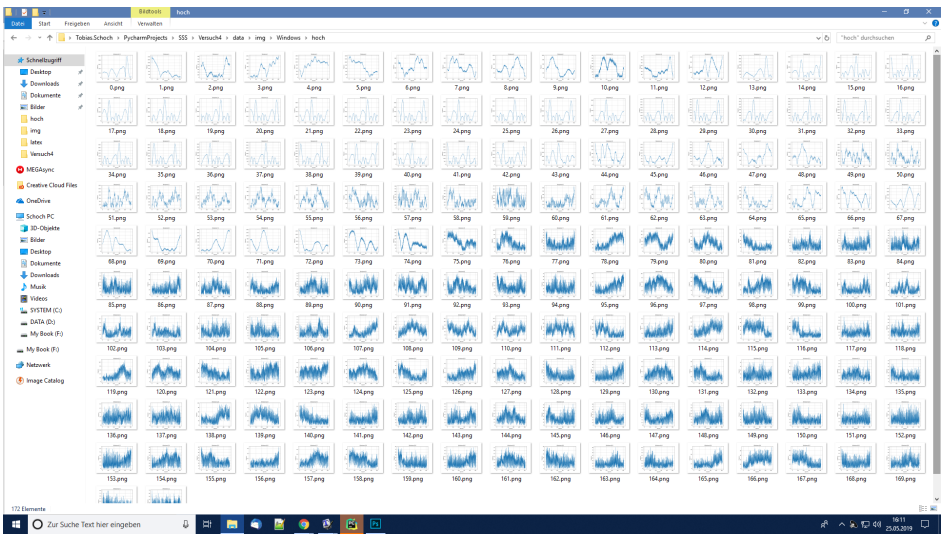


Abbildung 2.7: Aufnahme mit Triggerfunktion des Wortes rechts

2.4 Interpretation

3

Versuch 2

3.1 Fragestellung, Messprinzip, Aufbau, Messmittel

3.2 Messwerte

3.3 Auswertung

3.4 Interpretation

Anhang

A.1 Quellcode

A.1.1 Quellcode Versuch 1

```
import pyaudio
import numpy
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 = numpy.fromstring(data, 'Int16');
numpy.save('aufgabe4/test.npy', decoded)

stream.stop_stream()
stream.close()
p.terminate()
```

Listing 4.1: Einlesen der Sprachaufnahme und ablegen des Signals in eine Numpy Datei

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

def time_axis(arr):
    return np.array(range(len(arr)))/44100

FORMAT = pyaudio.paInt16
SAMPLEFREQ = 44100
FRAMESIZE = 44100
NOFFRAMES = 2

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()

start = np.argmax(np.abs(decoded) > 0.05) - 1024
end = start + 44100
triggered = decoded[start:end]
triggered = np.concatenate((triggered, [0]*(44100 - end - start)))

np.savetxt("aufgabe4/tief_5_" + str(int(time.time())) + ".npy", triggered)
```

Listing 4.2: Einlesen einer Sprachaufnahme mit Aktivierung durch Triggerung

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

# Einlesen der .csv Datei
data = np.load('data/test.npy')
freq = np.zeros(225280)

# Darstellung des Amplitudenspektrums
plt.plot(data)
plt.grid()
plt.xlabel('Zeit')
plt.ylabel('Amplitude')
plt.savefig('data/img/testamp.png')
plt.show()

# Einlesen der .csv Datei
data2 = np.load('data/rechts2.npy')

# Darstellung des Amplitudenspektrums
plt.plot(data2)
plt.grid()
plt.xlabel('Zeit')
plt.ylabel('Amplitude')
plt.savefig('data/img/rechtsamp.png')
plt.show()

# Darstellung des Amplitudenspektrums
plt.plot(data)
plt.grid()
plt.xlabel('Zeit')
plt.ylabel('Amplitude')
plt.xlim(50000, 75000)
plt.savefig('data/img/testamp2.png')
plt.show()

# Der zweite Wert wird absolut minus den ersten absoluten wert gerechnet um später den Wert
difference = 2 / 225280
# Die zweite Spalte der .csv Datei wird Fouriertransformiert
fourier = np.fft.fft(data[:225280])
# Die Fouriertransformierte Frequenz wird absolutiert, so dass kein negativer Wert mehr vorzufinden ist
spektrum = np.abs(fourier)
```

```
# Formel um die Anzahl der Schwingungen in die Freuquenz umzurechnen − f = n / (M ∗ t)
for x in range(0, 225280, 1):
    freq[x] = (x / (difference ∗ 225280))

# Darstellung des Amplitudenspektrums
plt.plot(freq, spektrum)
plt.grid()
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude in V')
plt.xlim(0, 60000)
plt.savefig('data/img/testspektrum1.png')
plt.show()

# Darstellung des Amplitudenspektrums
plt.plot(freq, spektrum)
plt.grid()
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude in V')
plt.xlim(0, 35000)
plt.savefig('data/img/testspektrum2.png')
plt.show()

# Darstellung des Amplitudenspektrums
plt.plot(freq, spektrum)
plt.grid()
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude in V')
plt.xlim(0, 1000)
plt.savefig('data/img/testspektrum3.png')
plt.show()
```

Listing 4.3: Amplitudenspektrum und Ausgabe von Plots

```
from scipy import signal

import matplotlib.pyplot as plt
import numpy as np

# Einlesen der .csv Datei
data = np.load('data/test.npy')
window = np.zeros((879, 512))
z = 256
gaussianwindow = signal.windows.gaussian(512, std=4)
for y in range(0, 879):
    z = z - 256
    for x in range(0, 512):
        window[y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
        z = z + 1
    # plt.plot(window[y])
    # plt.title('Windownr' + str(y+1))
    # plt.xlabel('Signalnr.')
    # plt.ylabel('Frequenz')
    # plt.grid(True)
    # plt.savefig('data/img/' + str(y) + '.png')
    # plt.show()

# Darstellung des Amplitudenspektrums
plt.plot(gaussianwindow)
plt.grid(True)
plt.xlabel('Signalnr.')
plt.ylabel('Frequenz')
plt.savefig('data/img/gauss.png')
plt.show()

# Darstellung des Amplitudenspektrums
plt.plot(window)
plt.grid(True)
plt.xlabel('Signalnr.')
plt.ylabel('Frequenz')
plt.savefig('data/img/Alle.png')
plt.show()

for y in range(0, 879):
    for x in range(0, 512):
```

```
42          window[y][x] = window[y][x] * gaussianwindow[x]
43      window[y] = np.abs(np.fft.fft(window[y]))
44      window[y] = np.mean(window[y])
45
46  plt.plot(window, 'r')
47  plt.grid(True)
48  plt.xlabel('Signalnr.')
49  plt.ylabel('Frequenz')
50  plt.savefig('data/img/AlleRichtig.png')
51  plt.show()
```

Listing 4.4: Windowing und Ausgabe von Plots bzw. Windows

A.1.2 Quellcode Versuch 2

```
from scipy import signal

import matplotlib.pyplot as plt
import numpy as np
import scipy.stats

# Einlesen der .csv Datei

num = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
    "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
numm = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
    "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
nummm = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
    "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
capital = ["hoch", "tief", "links", "rechts"]
capital2 = ["hoch", "tief", "links", "rechts"]
gaussianwindow = signal.windows.gaussian(512, std=4)

# Darstellung des Amplitudenspektrums
plt.plot(gaussianwindow)
plt.grid(True)
plt.xlabel('Signalnr.')
plt.ylabel('Frequenz')
plt.savefig('data/img/gauss.png')
plt.show()

for a in range(0, 20):
    data = np.load('data/' + str(num[a]) + '.npy')
    num[a] = np.zeros((171, 512))
    z = 256

    for y in range(0, 171):
        z = z - 256
        for x in range(0, 512):
            num[a][y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
            z = z + 1
        # plt.plot(num[a][y])
        # plt.title('Windownr' + str(y+1+(a*171)))
        # plt.xlabel('Signalnr.')
        # plt.ylabel('Frequenz')
```

```
41        # plt.grid(True)
42        # plt.savefig('data/img/' + str(y+1+(a*171)) + '.png')
43        # plt.show()
44
45    for y in range(0, 171):
46        for x in range(0, 512):
47            num[a][y, x] = num[a][y, x] * gaussianwindow[x]
48        num[a][y] = np.abs(np.fft.fft(num[a][y]))
49        num[a][y] = np.mean(num[a][y])
50
51    plt.plot(num[a], 'r')
52    plt.title(str(nummm[a]))
53    plt.grid(True)
54    plt.xlabel('Signalnr.')
55    plt.ylabel('Frequenz')
56    plt.savefig('data/img/' + numm[a] + 'AlleRichtig.png')
57    plt.show()
58
59 for z in range(0, 4):
60    capital[z] = np.zeros((171, 512))
61    for y in range(0, 171):
62        for x in range(0, 512):
63            if (z == 0):
64                capital[z][y, x] = (num[0][y, x] + num[1][y, x] + num[2][y, x] + num[3][y, x] + num[4][y, x]) / 4
65            elif (z == 1):
66                capital[z][y, x] = (num[5][y, x] + num[6][y, x] + num[7][y, x] + num[8][y, x] + num[9][y, x]) / 4
67            elif (z == 2):
68                capital[z][y, x] = (num[10][y, x] + num[11][y, x] + num[12][y, x] + num[13][y, x] + num[14][y, x]) / 4
69            elif (z == 3):
70                capital[z][y, x] = (num[15][y, x] + num[16][y, x] + num[17][y, x] + num[18][y, x] + num[19][y, x]) / 4
71
72    plt.plot(capital[z], 'r')
73    plt.grid(True)
74    plt.xlabel('Signalnr.')
75    plt.ylabel('Frequenz')
76    plt.savefig('data/img/' + capital2[z] + 'Average.png')
77    plt.show()
```

Listing 4.5: Windowing und Mittelung der Spektren

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

# Einlesen der .csv Datei

num = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
       "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
numm = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
        "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
capital = ["hoch", "tief", "links", "rechts"]
capital2 = ["hoch", "tief", "links", "rechts"]

gaussianwindow = signal.windows.gaussian(512, std=4)

for a in range(0, 20):
    data = np.load('data/' + str(num[a]) + '.npy')
    num[a] = np.zeros((171, 512))
    z = 256

    for y in range(0, 171):
        z = z - 256
        for x in range(0, 512):
            num[a][y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
            z = z + 1

    for y in range(0, 171):
        for x in range(0, 512):
            num[a][y, x] = num[a][y, x] * gaussianwindow[x]
        num[a][y] = np.abs(np.fft.fft(num[a][y]))
        num[a][y] = np.mean(num[a][y])

for z in range(0, 4):
    capital[z] = np.zeros((171, 512))
    for y in range(0, 171):
        for x in range(0, 512):
            if (z == 0):
                capital[z][y, x] = (num[0][y, x] + num[1][y, x] + num[2][y, x] + num[3][y, x] + num[4][y, x]) / 4
            elif (z == 1):
                capital[z][y, x] = (num[5][y, x] + num[6][y, x] + num[7][y, x] + num[8][y, x] + num[9][y, x]) / 4
```

```
42            elif (z == 2):
43                capital[z][y, x] = (num[10][y, x] + num[11][y, x] + num[12][y, x] + num[13][y, x] + num[14][y, x]) / 4
44            elif (z == 3):
45                capital[z][y, x] = (num[15][y, x] + num[16][y, x] + num[17][y, x] + num[18][y, x] + num[19][y, x]) / 4
46        capital[z] = capital[z].ravel()
47
48    for x in range(0, 20):
49        num[x] = num[x].ravel()
50
51    r, p = scipy.stats.pearsonr(num[0], capital[0])
52    print("r:", r, "p:", p)
```

Listing 4.6: Windowing und Bravais-Pearson Methode

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

# Einlesen der .csv Datei

num = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
      "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
anderer = ["ahoch1", "ahoch2", "ahoch3", "ahoch4", "ahoch5", "atief1", "atief2", "atief3", "atief4", "atief5", "alinks1", "alinks2"
       "alinks3", "alinks4", "alinks5", "arechts1", "arechts2", "arechts3", "arechts4", "arechts5"]
moi = ["mhoch1", "mhoch2", "mhoch3", "mhoch4", "mhoch5", "mtief1", "mtief2", "mtief3", "mtief4", "mtief5", "mlinks1", "ml
      "mlinks3", "mlinks4", "mlinks5", "mrechts1", "mrechts2", "mrechts3", "mrechts4", "arechts5"]
capital = ["hoch", "tief", "links", "rechts"]
capital2 = ["hoch", "tief", "links", "rechts"]

gaussianwindow = signal.windows.gaussian(512, std=4)

for a in range(0, 20):
    data = np.load('data/' + str(num[a]) + '.npy')
    data2 = np.load('data/' + str(anderer[a]) + '.npy')
    data3 = np.load('data/' + str(moi[a]) + '.npy')
    num[a] = np.zeros((171, 512))
    moi[a] = np.zeros((171, 512))
    anderer[a] = np.zeros((171, 512))
    z = 256

    for y in range(0, 171):
        z = z - 256
        for x in range(0, 512):
            num[a][y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
            anderer[a][y, x] = np.mean(np.abs(np.fft.fft(data2[z] * gaussianwindow)))
            moi[a][y, x] = np.mean(np.abs(np.fft.fft(data3[z] * gaussianwindow)))
            z = z + 1

    for y in range(0, 171):
        for x in range(0, 512):
            num[a][y, x] = num[a][y, x] * gaussianwindow[x]
            anderer[a][y, x] = anderer[a][y, x] * gaussianwindow[x]
            moi[a][y, x] = moi[a][y, x] * gaussianwindow[x]
        num[a][y] = np.abs(np.fft.fft(num[a][y]))
```

```
        anderer[a][y] = np.abs(np.fft.fft(anderer[a][y]))
        moi[a][y] = np.abs(np.fft.fft(moi[a][y]))
        num[a][y] = np.mean(num[a][y])
        anderer[a][y] = np.mean(anderer[a][y])
        moi[a][y] = np.mean(moi[a][y])

for z in range(0, 4):
    capital[z] = np.zeros((171, 512))
    for y in range(0, 171):
        for x in range(0, 512):
            if (z == 0):
                capital[z][y, x] = (num[0][y, x] + num[1][y, x] + num[2][y, x] + num[3][y, x] + num[4][y, x]) / 4
            elif (z == 1):
                capital[z][y, x] = (num[5][y, x] + num[6][y, x] + num[7][y, x] + num[8][y, x] + num[9][y, x]) / 4
            elif (z == 2):
                capital[z][y, x] = (num[10][y, x] + num[11][y, x] + num[12][y, x] + num[13][y, x] + num[14][y, x]) / 4
            elif (z == 3):
                capital[z][y, x] = (num[15][y, x] + num[16][y, x] + num[17][y, x] + num[18][y, x] + num[19][y, x]) / 4
    capital[z] = capital[z].ravel()

for x in range(0, 20):
    num[x] = num[x].ravel()
    moi[x] = moi[x].ravel()
    anderer[x] = anderer[x].ravel()


r, p = scipy.stats.pearsonr(num[0], num[0])
print("num−num r:", r, "p:", p)

r, p = scipy.stats.pearsonr(capital[0], num[0])
print("capital−num r:", r, "p:", p)

r, p = scipy.stats.pearsonr(capital[0], moi[0])
print("capital−moi r:", r, "p:", p)

r, p = scipy.stats.pearsonr(capital[0], anderer[0])
print("capital−anderer r:", r, "p:", p)

r, p = scipy.stats.pearsonr(num[0], moi[0])
print("num−moi1 r:", r, "p:", p)
r, p = scipy.stats.pearsonr(num[1], moi[1])
print("num−moi2 r:", r, "p:", p)
```

```
84 r, p = scipy.stats.pearsonr(num[2], moi[2])
85 print("num-moi3 r:", r, "p:", p)
86
87
88 r, p = scipy.stats.pearsonr(num[0], anderer[0])
89 print("num-anderer1 r:", r, "p:", p)
90 r, p = scipy.stats.pearsonr(num[1], anderer[1])
91 print("num-anderer2 r:", r, "p:", p)
92 r, p = scipy.stats.pearsonr(num[2], anderer[2])
93 print("num-anderer3 r:", r, "p:", p)
```

Listing 4.7: Bravais-Pearson Methode mit Ausgabe der Korrelation