



**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

<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>IV</b>
<b>Listingverzeichnis</b>	<b>V</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Versuch 1</b>	<b>2</b>
2.1 Fragestellung, Messprinzip, Aufbau, Messmittel . . . . .	2
2.2 Messwerte . . . . .	5
2.3 Auswertung . . . . .	6
2.4 Interpretation . . . . .	7
<b>3 Versuch 2</b>	<b>8</b>
3.1 Fragestellung, Messprinzip, Aufbau, Messmittel . . . . .	8
3.2 Messwerte . . . . .	8
3.3 Auswertung . . . . .	8
3.4 Interpretation . . . . .	8
<b>Anhang</b>	<b>9</b>
A.1 Quellcode . . . . .	9
A.1.1 Quellcode Versuch 1 . . . . .	9
A.1.2 Quellcode Versuch 2 . . . . .	15

# Abbildungsverzeichnis

2.1	Aufnahme mit Triggerfunktion des Wortes rechts . . . . .	3
2.2	Die aufgenommene Sprachaufnahme visualisiert mit Python . . . . .	5
2.2a	Sprachaufnahme des Wortes Test . . . . .	5
2.2b	Dieselbe Sprachaufnahme mit kurzer Zeitachse . . . . .	5
2.3	Aufnahme mit Triggerfunktion des Wortes rechts . . . . .	5
2.4	Das Amplitudenspektrum mit der dazugehörigen Frequenz . . . . .	6
2.4a	Amplitudenspektrum der Sprachaufnahme . . . . .	6
2.4b	Amplitudenspektrum mit geringerer Frequenz . . . . .	6
2.5	Aufnahme mit Triggerfunktion des Wortes rechts . . . . .	6
2.6	Aufnahme mit Triggerfunktion des Wortes rechts . . . . .	7
2.7	Aufnahme mit Triggerfunktion des Wortes rechts . . . . .	7

# **Tabellenverzeichnis**

# Listingverzeichnis

4.1	Einlesen der Sprachaufnahme und ablegen des Signals in eine Numpy Datei	9
4.2	Einlesen einer Sprachaufnahme mit Aktivierung durch Triggerung . . . . .	10
4.3	Amplitudenspektrum und Ausgabe von Plots . . . . .	11
4.4	Windowing und Ausgabe von Plots bzw. Windows . . . . .	13
4.5	Windowing und Mittelung der Spektren . . . . .	15
4.6	Windowing und Bravais-Pearson Methode . . . . .	17
4.7	Bravais-Pearson Methode mit Ausgabe der Korrelation . . . . .	19

# 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 Klinkestecker 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 aufgefüllt werden.

Mit dem Code Aus dem dritten Versuch können wir mit der Aufnahme das Amplitudenspektrum bestimmen. Dies stellen wir graphisch 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 Standardabweichung 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.



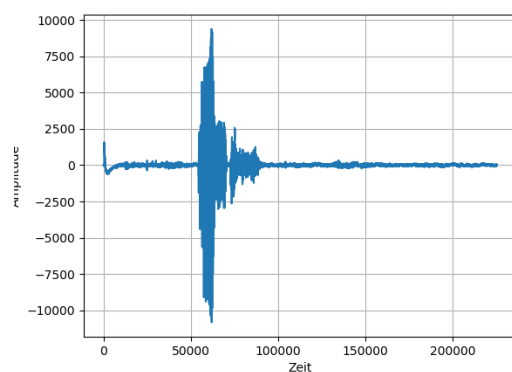
Abbildung 2.1: Aufnahme mit Triggerfunktion des Wortes rechts

**Messmittel:**

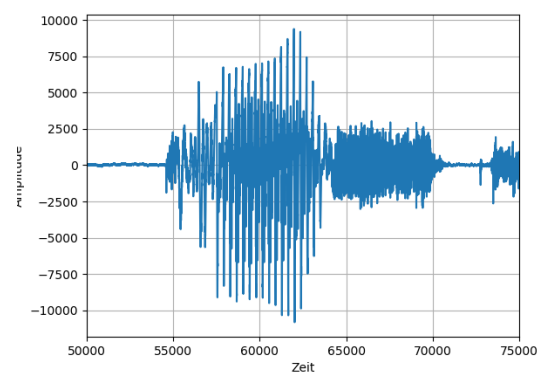
- Ein Mikrofon
- Ein Computer mit einer Python IDE

## 2.2 Messwerte

Mit einem Mikrofon das an den Computer angeschlossen ist, sehen wir die mit dem Python-skript 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.

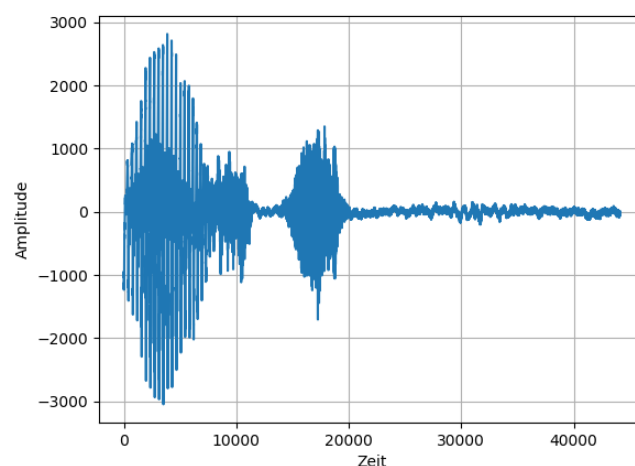
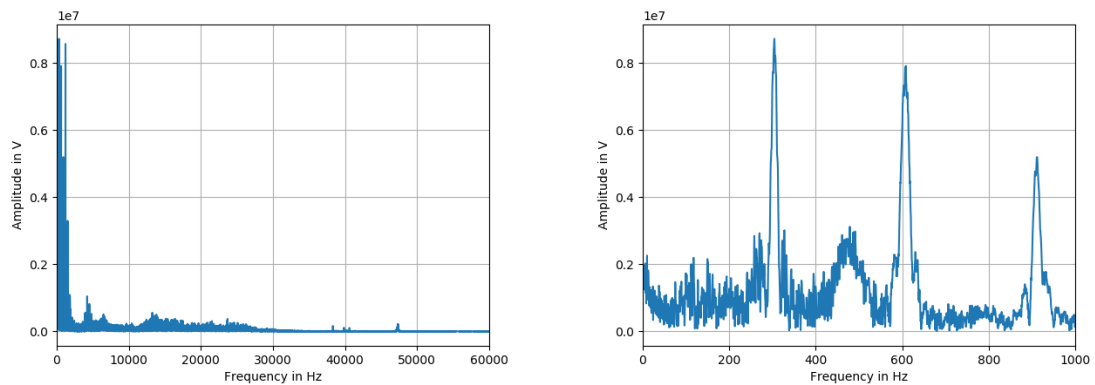


Abbildung 2.3: Aufnahme mit Triggerfunktion des Wortes rechts

## 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.

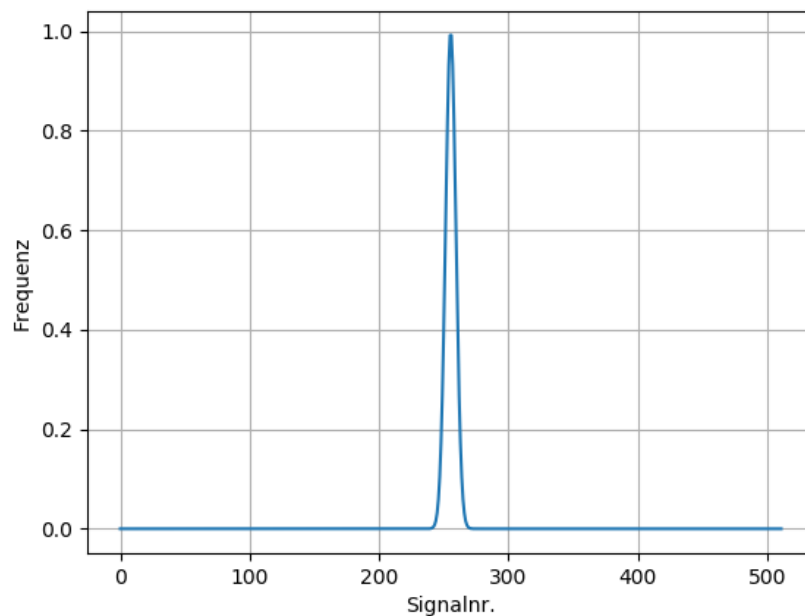


Abbildung 2.5: Aufnahme mit Triggerfunktion des Wortes rechts

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

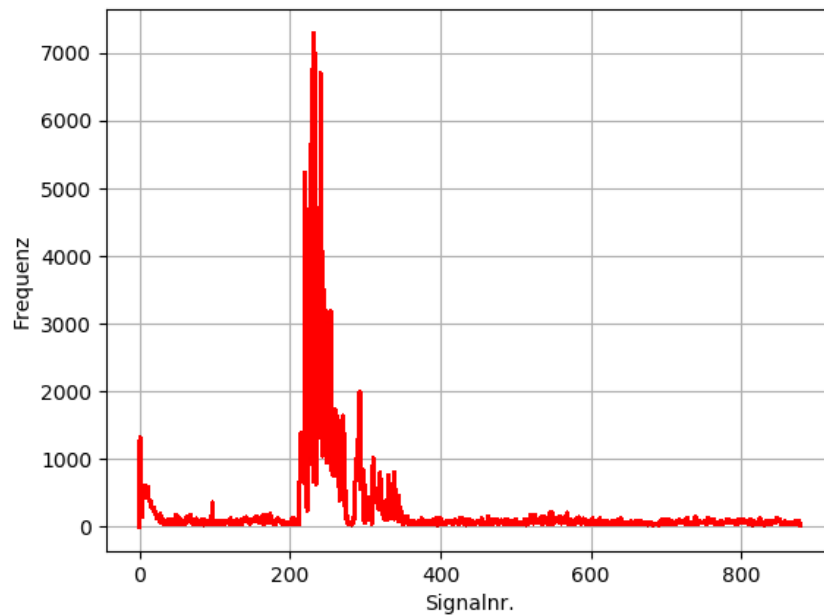


Abbildung 2.6: Aufnahme mit Triggerfunktion des Wortes rechts

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

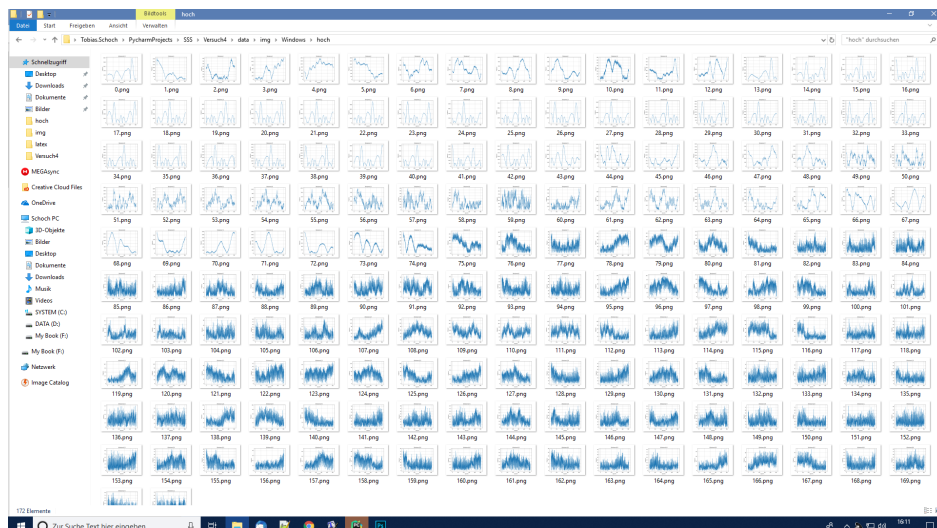


Abbildung 2.7: Aufnahme mit Triggerfunktion des Wortes rechts

## **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

```
1 import pyaudio
2 import numpy
3 import matplotlib.pyplot as plt
4
5 FORMAT = pyaudio.paInt16
6 SAMPLEFREQ = 44100
7 FRAMESIZE = 1024
8 NOFFRAMES = 220
9 p = pyaudio.PyAudio()
10 print('running')
11
12 stream = p.open(format=FORMAT,
13                 channels=1,
14                 rate=SAMPLEFREQ,
15                 input=True,
16                 frames_per_buffer=FRAMESIZE)
17 data = stream.read(NOFFRAMES*FRAMESIZE)
18 decoded = numpy.fromstring(data, 'Int16');
19 numpy.save('aufgabe4/test.npy', decoded)
20
21 stream.stop_stream()
22 stream.close()
23 p.terminate()
```

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



```

1 import pyaudio
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import time
5
6 def time_axis(arr):
7     return np.array(range(len(arr)))/44100
8
9 FORMAT = pyaudio.paInt16
10 SAMPLEFREQ = 44100
11 FRAMESIZE = 44100
12 NOFFRAMES = 2
13
14 p = pyaudio.PyAudio()
15 print('running')
16 stream = p.open(format=FORMAT, channels=1, rate=SAMPLEFREQ,
17                 input=True, frames_per_buffer=FRAMESIZE)
18 data = stream.read(NOFFRAMES * FRAMESIZE)
19 decoded = np.fromstring(data, 'Int16') / ((2**15)/2-1)
20 stream.stop_stream()
21 stream.close()
22 p.terminate()
23
24 start = np.argmax(np.abs(decoded) > 0.05) - 1024
25 end = start + 44100
26 triggered = decoded[start:end]
27 triggered = np.concatenate((triggered, [0]*(44100 - end - start)))
28
29 np.savetxt("aufgabe4/tief_5_" + str(int(time.time())) + ".npy", triggered)

```

Listing 4.2: Einlesen einer Sprachaufnahme mit Aktivierung durch Triggerung

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Einlesen der .csv Datei
5 data = np.load('data/test.npy')
6 freq = np.zeros(225280)
7
8 # Darstellung des Amplitudenspektrums
9 plt.plot(data)
10 plt.grid()
11 plt.xlabel('Zeit')
12 plt.ylabel('Amplitude')
13 plt.savefig('data/img/testamp.png')
14 plt.show()
15
16 # Einlesen der .csv Datei
17 data2 = np.load('data/rechts2.npy')
18
19 # Darstellung des Amplitudenspektrums
20 plt.plot(data2)
21 plt.grid()
22 plt.xlabel('Zeit')
23 plt.ylabel('Amplitude')
24 plt.savefig('data/img/rechtsamp.png')
25 plt.show()
26
27 # Darstellung des Amplitudenspektrums
28 plt.plot(data)
29 plt.grid()
30 plt.xlabel('Zeit')
31 plt.ylabel('Amplitude')
32 plt.xlim(50000, 75000)
33 plt.savefig('data/img/testamp2.png')
34 plt.show()
35
36 # Der zweite Wert wird absolut minus den ersten absoluten wert gerechnet um später den Wert
37 difference = 2 / 225280
38 # Die zweite Spalte der .csv Datei wird Fouriertransformiert
39 fourier = np.fft.fft(data[:225280])
40 # Die Fouriertransformierte Frequenz wird absolutiert, so dass kein negativer Wert mehr vorzufinden ist
41 spektrum = np.abs(fourier)

```

```

42 # Formel um die Anzahl der Schwingungen in die Frequenz umzurechnen —  $f = n / (M * t)$ 
43 for x in range(0, 225280, 1):
44     freq[x] = (x / (difference * 225280))
45
46 # Darstellung des Amplitudenspektrums
47 plt.plot(freq, spektrum)
48 plt.grid()
49 plt.xlabel('Frequency in Hz')
50 plt.ylabel('Amplitude in V')
51 plt.xlim(0, 60000)
52 plt.savefig('data/img/testspektrum1.png')
53 plt.show()
54
55 # Darstellung des Amplitudenspektrums
56 plt.plot(freq, spektrum)
57 plt.grid()
58 plt.xlabel('Frequency in Hz')
59 plt.ylabel('Amplitude in V')
60 plt.xlim(0, 35000)
61 plt.savefig('data/img/testspektrum2.png')
62 plt.show()
63
64 # Darstellung des Amplitudenspektrums
65 plt.plot(freq, spektrum)
66 plt.grid()
67 plt.xlabel('Frequency in Hz')
68 plt.ylabel('Amplitude in V')
69 plt.xlim(0, 1000)
70 plt.savefig('data/img/testspektrum3.png')
71 plt.show()

```

Listing 4.3: Amplitudenspektrum und Ausgabe von Plots

```

1 from scipy import signal
2
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 # Einlesen der .csv Datei
7 data = np.load('data/test.npy')
8 window = np.zeros((879, 512))
9 z = 256
10 gaussianwindow = signal.windows.gaussian(512, std=4)
11 for y in range(0, 879):
12     z = z - 256
13     for x in range(0, 512):
14         window[y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
15         z = z + 1
16     # plt.plot(window[y])
17     # plt.title('Windownr' + str(y+1))
18     # plt.xlabel('Signalnr.')
19     # plt.ylabel('Frequenz')
20     # plt.grid(True)
21     # plt.savefig('data/img/' + str(y) + '.png')
22     # plt.show()
23
24 # Darstellung des Amplitudenspektrums
25 plt.plot(gaussianwindow)
26 plt.grid(True)
27 plt.xlabel('Signalnr.')
28 plt.ylabel('Frequenz')
29 plt.savefig('data/img/gauss.png')
30 plt.show()
31
32 # Darstellung des Amplitudenspektrums
33 plt.plot(window)
34 plt.grid(True)
35 plt.xlabel('Signalnr.')
36 plt.ylabel('Frequenz')
37 plt.savefig('data/img/Alle.png')
38 plt.show()
39
40 for y in range(0, 879):
41     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

```
1 from scipy import signal
2
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import scipy.stats
6
7 # Einlesen der .csv Datei
8
9 num = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
10        "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
11 numm = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
12         "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
13 nummm = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
14          "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
15 capital = ["hoch", "tief", "links", "rechts"]
16 capital2 = ["hoch", "tief", "links", "rechts"]
17 gaussianwindow = signal.windows.gaussian(512, std=4)
18
19 # Darstellung des Amplitudenspektrums
20 plt.plot(gaussianwindow)
21 plt.grid(True)
22 plt.xlabel('Signalnr.')
23 plt.ylabel('Frequenz')
24 plt.savefig('data/img/gauss.png')
25 plt.show()
26
27 for a in range(0, 20):
28     data = np.load('data/' + str(num[a]) + '.npy')
29     num[a] = np.zeros((171, 512))
30     z = 256
31
32     for y in range(0, 171):
33         z = z - 256
34         for x in range(0, 512):
35             num[a][y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
36             z = z + 1
37         # plt.plot(num[a][y])
38         # plt.title('Windownr' + str(y+1+(a*171)))
39         # plt.xlabel('Signalnr.')
40         # 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

```

1 from scipy import signal
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import scipy.stats
5
6 # Einlesen der .csv Datei
7
8 num = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
9        "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
10 numm = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
11         "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
12 capital = ["hoch", "tief", "links", "rechts"]
13 capital2 = ["hoch", "tief", "links", "rechts"]
14
15 gaussianwindow = signal.windows.gaussian(512, std=4)
16
17 for a in range(0, 20):
18     data = np.load('data/' + str(num[a]) + '.npy')
19     num[a] = np.zeros((171, 512))
20     z = 256
21
22     for y in range(0, 171):
23         z = z - 256
24         for x in range(0, 512):
25             num[a][y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
26             z = z + 1
27
28     for y in range(0, 171):
29         for x in range(0, 512):
30             num[a][y, x] = num[a][y, x] * gaussianwindow[x]
31             num[a][y] = np.abs(np.fft.fft(num[a][y]))
32             num[a][y] = np.mean(num[a][y])
33
34     for z in range(0, 4):
35         capital[z] = np.zeros((171, 512))
36         for y in range(0, 171):
37             for x in range(0, 512):
38                 if (z == 0):
39                     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
40                 elif (z == 1):
41                     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

```

1 from scipy import signal
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import scipy.stats
5
6 # Einlesen der .csv Datei
7
8 num = ["hoch1", "hoch2", "hoch3", "hoch4", "hoch5", "tief1", "tief2", "tief3", "tief4", "tief5", "links1", "links2",
9        "links3", "links4", "links5", "rechts1", "rechts2", "rechts3", "rechts4", "rechts5"]
10 anderer = ["ahoch1", "ahoch2", "ahoch3", "ahoch4", "ahoch5", "atief1", "atief2", "atief3", "atief4", "atief5", "alinks1", "alinks2",
11            "alinks3", "alinks4", "alinks5", "arechts1", "arechts2", "arechts3", "arechts4", "arechts5"]
12 moi = ["mhoch1", "mhoch2", "mhoch3", "mhoch4", "mhoch5", "mtief1", "mtief2", "mtief3", "mtief4", "mtief5", "mlinks1", "mlinks2",
13        "mlinks3", "mlinks4", "mlinks5", "mrechts1", "mrechts2", "mrechts3", "mrechts4", "mrechts5"]
14 capital = ["hoch", "tief", "links", "rechts"]
15 capital2 = ["hoch", "tief", "links", "rechts"]
16
17 gaussianwindow = signal.windows.gaussian(512, std=4)
18
19 for a in range(0, 20):
20     data = np.load('data/' + str(num[a]) + '.npy')
21     data2 = np.load('data/' + str(anderer[a]) + '.npy')
22     data3 = np.load('data/' + str(moi[a]) + '.npy')
23     num[a] = np.zeros((171, 512))
24     moi[a] = np.zeros((171, 512))
25     anderer[a] = np.zeros((171, 512))
26     z = 256
27
28     for y in range(0, 171):
29         z = z - 256
30         for x in range(0, 512):
31             num[a][y, x] = np.mean(np.abs(np.fft.fft(data[z] * gaussianwindow)))
32             anderer[a][y, x] = np.mean(np.abs(np.fft.fft(data2[z] * gaussianwindow)))
33             moi[a][y, x] = np.mean(np.abs(np.fft.fft(data3[z] * gaussianwindow)))
34             z = z + 1
35
36     for y in range(0, 171):
37         for x in range(0, 512):
38             num[a][y, x] = num[a][y, x] * gaussianwindow[x]
39             anderer[a][y, x] = anderer[a][y, x] * gaussianwindow[x]
40             moi[a][y, x] = moi[a][y, x] * gaussianwindow[x]
41     num[a][y] = np.abs(np.fft.fft(num[a][y]))

```

```

42     anderer[a][y] = np.abs(np.fft.fft(anderer[a][y]))
43     moi[a][y] = np.abs(np.fft.fft(moi[a][y]))
44     num[a][y] = np.mean(num[a][y])
45     anderer[a][y] = np.mean(anderer[a][y])
46     moi[a][y] = np.mean(moi[a][y])
47
48 for z in range(0, 4):
49     capital[z] = np.zeros((171, 512))
50     for y in range(0, 171):
51         for x in range(0, 512):
52             if (z == 0):
53                 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
54             elif (z == 1):
55                 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
56             elif (z == 2):
57                 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
58             elif (z == 3):
59                 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
60     capital[z] = capital[z].ravel()
61
62 for x in range(0, 20):
63     num[x] = num[x].ravel()
64     moi[x] = moi[x].ravel()
65     anderer[x] = anderer[x].ravel()
66
67
68 r, p = scipy.stats.pearsonr(num[0], num[0])
69 print("num—num r:", r, "p:", p)
70
71 r, p = scipy.stats.pearsonr(capital[0], num[0])
72 print("capital—num r:", r, "p:", p)
73
74 r, p = scipy.stats.pearsonr(capital[0], moi[0])
75 print("capital—moi r:", r, "p:", p)
76
77 r, p = scipy.stats.pearsonr(capital[0], anderer[0])
78 print("capital—anderer r:", r, "p:", p)
79
80 r, p = scipy.stats.pearsonr(num[0], moi[0])
81 print("num—moi1 r:", r, "p:", p)
82 r, p = scipy.stats.pearsonr(num[1], moi[1])
83 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