Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
ISC Institute of Signal Processing and Wireless Communications

Soundkarte mit Python

Version 1.0, 2.7.2024

Zürcher Hochschule für Angewandte Wissenschaften (ZHAW)
Institute of Signal Processing and Wireless Communications (ISC)
Technikumstrasse 71
8401 Winterthur

Inhalt

1 Einführung

Die Sisy und DSV Praktikas werden von MATLAB auf Python umgestellt. Dieses Dokument ist eine Einführung, wie man die Soundkarte auch in Python nutzen kann.

1.1 Setup

1.1.1 Installation

Nutze das Package «sounddevice»: https://python-sounddevice.readthedocs.io/en/0.4.7/

Sounddevice kann mit folgenden Befehlen von der Konsole aus, installiert werden:

- PIP:
  - `pip install sounddevice==0.4.7`
- Conda
  - `conda install -c conda-forge python-sounddevice`

2 Nutzung der Soundkarte in Python

2.1 Selektion der Soundkarte

Python code:

```
import sounddevice as sd
import numpy as np
import matplotlib.pyplot as plt
print("connected devices")
print(sd.query_devices())
```

Output: → Alle Audio-Devices werden angezeigt, egal ob Input oder Output.

```
connected devices
   0 Microsoft Sound Mapper - Input, MME (2 in, 0 out)
>  1 Kopfhörermikrofon (Plantronics , MME (2 in, 0 out)
   2 Mikrofon (Scarlett 2i2 USB), MME (2 in, 0 out)
   3 Mikrofon (HD Webcam C525), MME (2 in, 0 out)
   4 Microsoft Sound Mapper - Output, MME (0 in, 2 out)
<  5 Lautsprecher (Scarlett 2i2 USB), MME (0 in, 2 out)
   6 Lautsprecher (Realtek Audio), MME (0 in, 2 out)
   7 DELL U2415 (2- NVIDIA High Defi, MME (0 in, 2 out)
```

➔ 1 Kopfhörermikrofon wird als Mikrofon verwendet
➔ 5 Lautsprecher (Scarlett 2i2 USB) wird als Lautsprecher verwendet

Automatische Auswahl des richtigen Mikrofons und Lautsprechers (wenn mehrere API's vorhanden sind, wird die erste API in der Liste verwendet):

```
info = sd.query_devices()
input_devID, output_devID = None, None
for dev in info:
    if 'Focusrite' in dev['name'] or 'Scarlett' in dev['name']:
        if dev['max_input_channels'] > 0 and input_devID is None:
            input_devID = dev['index']
        if dev['max_output_channels'] > 0 and output_devID is None:
            output_devID = dev['index']
if input_devID is None or output_devID is None:
    print("USB audio interface not found. Please connect the appropriate device.")
    exit()
```

Wenn nicht die richtige Soundkarte verbunden ist, wird folgende Fehlermeldung auf der Konsole ausgegeben:

USB audio interface not found. Please connect the appropriate device.

2.2 Einlesen und Abspielen von Audios

2.2.1 Separat Einlesen und Abspielen

```
# record and play back one second of audio:
fs = 48e3
tRecord = 1 # [s]
x = sd.rec(frames = int(fs*tRecord), samplerate=fs, channels=2,
device=input_devID, blocking=True)[:,0] # select channel 0

# normalized playback
sd.play(x/np.max(np.abs(x)), samplerate=fs, blocking=True, device=output_devID)
```

Alternativ kann auch mit `sd.wait()` auf die Soundkarte gewartet werden:

```
# record and play back one second of audio:
fs = 48e3
tRecord = 1 # [s]
x = sd.rec(frames = int(fs*tRecord), samplerate=fs, channels=2,
device=input_devID)[:,0] # select channel 0
sd.wait()

# normalized playback
sd.play(x/np.max(np.abs(x)), samplerate=fs, device=output_devID)
sd.wait()
```

2.2.2 Gleichzeitig Einlesen und Abspielen

```
fs = 48e3
t = np.arange(fs)/fs
f = 1e3
playSig = np.sin(2*np.pi*f*t)
sd.default.device = [input_devID,output_devID]
sd.default.channels = 2
recordSig = sd.playrec(playSig, samplerate = fs, blocking=True)
```

Alternativ kann auch mit `sd.wait()` auf die Soundkarte gewartet werden:

```
# record and play back one second of audio:
fs = 48e3
t = np.arange(fs)/fs
f = 1e3
playSig = np.sin(2*np.pi*f*t)
sd.default.device = [input_devID,output_devID]
sd.default.channels = 2
recordSig = sd.playrec(playSig, samplerate = fs)
sd.wait()
```

3 Korrelation

`np.correlate` ist sehr langsam für Korrelationen von langen Signalen.

Experiment: Autokorrelation einer PN-Sequenz “s” der Länge 262143:

```
import scipy.signal as sig
import numpy as np
M = 18            # Memory des LFSR
L = 2**M-1;       # Länge MLS- bzw. PN-Sequenz PN12 dauert 84ms
## Generierung des periodischen PN-Signals, senden und empfangen
s = np.zeros(L)
s[0:(M)] = np.ones(M) # seed bzw. Startwert Schieberegister
for m in range (M,L):
    s[m] = np.mod(s[m-18]+s[m-11],2);  # p(X) = 1+X11+X18

s = 1-2*s; # binary2bipolar: 0 => 1 und 1 => -1

y0 = np.convolve(np.flip(s),s)
y1 = np.correlate(s, s), mode='full')
y2 = sig.correlate(s, s)
y3 = sig.fftconvolve(np.flip(s), s)
```

Function	*Runtime [s]*
`np.convolve`	45
`np.correlate`	45
`sig.correlate`	**0.024**
`sig.fftconvolve`	**0.024**