Protokol řešení projektu v předmětu ISS

Vojtěch Šíma, xsimav01

4.1.2021

Úvod

Pro nahrání tónů a vět jsem využil programu Audacity. K udržení stejného tónu (v mém případě tón "a") jsem využil doporučené aplikace Fine Tuner.

Úkol 1

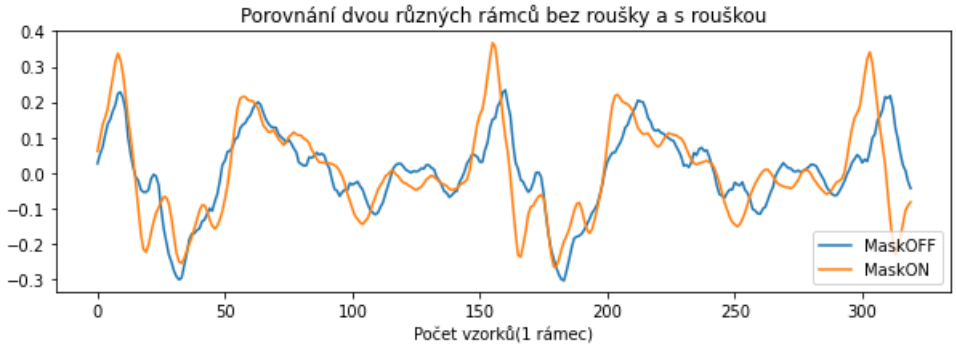
Název souboru	Délka[vzorky]	Délka[s]
maskoff_tone.wav	78762	4.92
maskon_tone.wav	76171	4.76

Úkol 2

Název souboru	Délka[vzorky]	Délka[s]
maskoff_sentence.wav	29210	1.83
maskon_sentence.wav	26834	1.68

Úkol 3

Extrahování vteřinových nahrávek jsem udělal ručně téže v programu Audacity (soubory on.wav a off.wav). Normalizace nebyla potřeba díky použití knihovny soundfile a k ustřednění jsem použil funkce np.mean z knihovny numpy.



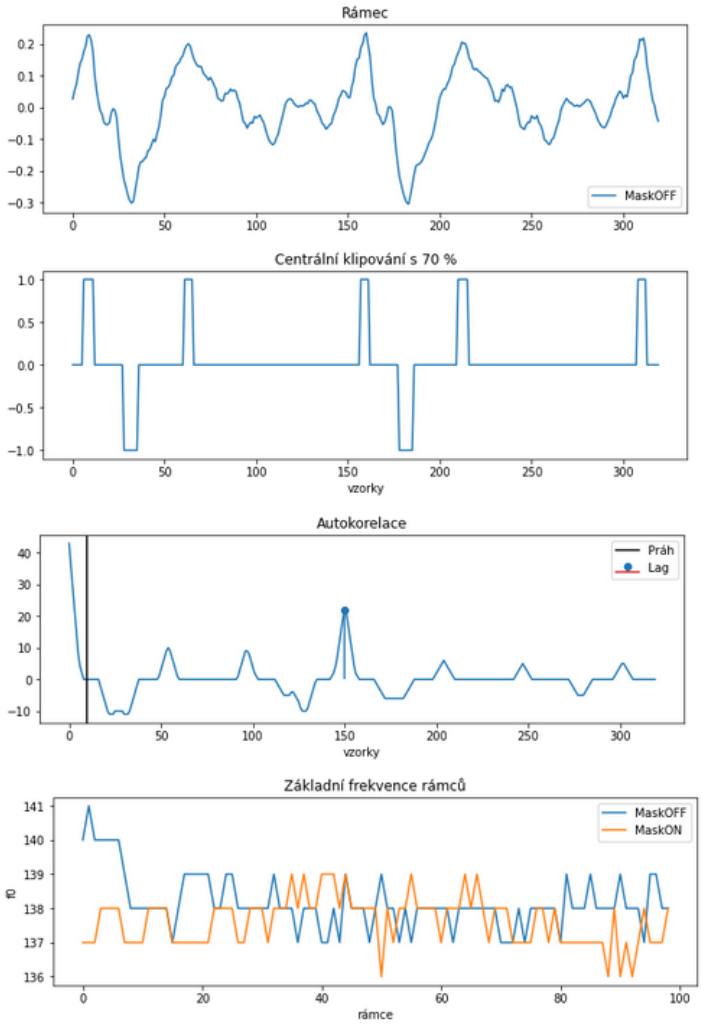
počet vzorků v jedné vteřině = 16000
počet rámců = 100 => 0,01
rámce se polovinou překrývají => 2
velikosti rámce ve vzorcích = 16000*0,01*2

Úkol 4

Pomocí vestavěných funkcí min() a max() si zjistíme extrémy a upravíme rámec na hodnota 0/1/-1. Následně provedeme autokorelaci, která je implementována pomocí dvou forů s různou velikostí (viz. obrázek), které zaručují, že se mezi sebou budou vždy násobit jiné vzorky. Střední hodnotu jsem počítal pomocí numpy.mean a rozptyl pomocí numpy.var.

```
for c in range(0,99): #pro všechny rámce
    for i in range(0,320): #cely ramec
        for j in range(0,320-i): #zkraceny ramec (posunut na zacatku i na konci o i)
            suma=suma+(array[c][j]*array[c][i+j])
            suma2=suma2+(array2[c][j]*array2[c][j+i])
        autocorr.append(suma)
        autocorr2.append(suma2)
        suma=0
        suma2=0
```

A)



B)

```
Střední hodnota bez roušky 138.16161616161617
Rozptyl bez roušky 0.6203448627691053
Střední hodnota s rouškou 137.62626262626262
Rozptyl s rouškou 0.4966840118355269
```

C)

Velikost změny f0 při chybě by se dala změnit větším počtem vzorků (například zdvojnásobením), což by vedlo k menším změnám f0.

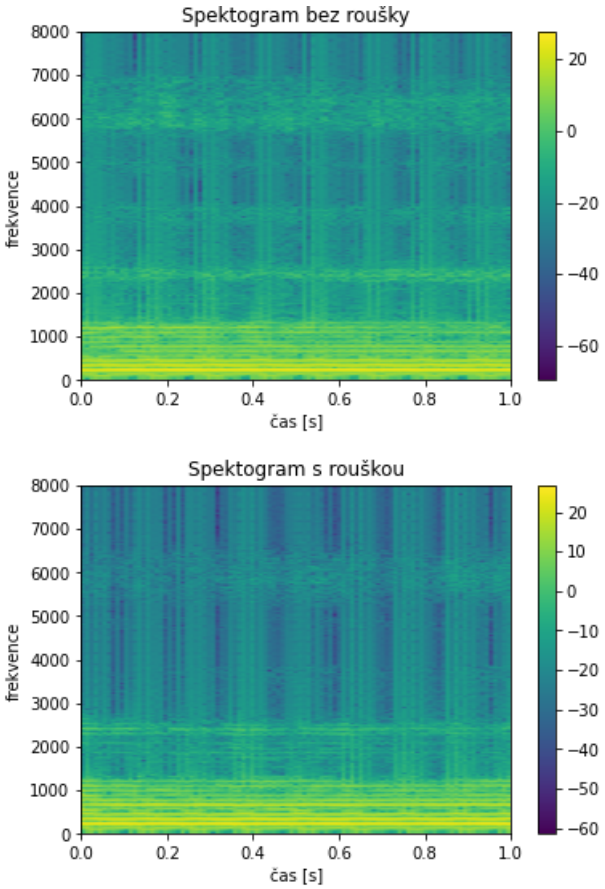
Úkol 5

Před aplikováním DFT bylo potřeba rozšířit rámce na velikost 1024 přidáním nul pomocí numpy.pad. Pak hned následovala DFT daného rámce, který byl následně zlogaritmován podle vzorce. Následně před vykreslením byly rámce zmenšeny na 512 hodnot a byl vykreslen spektogram pomocí imshow()

A)

```
def dft(x):
    N=x.shape[0]
    n=np.arange(N)
    k=n.reshape((N,1))
    SUM=np.exp(-2j*np.pi*k*n/N)
    return np.dot(SUM,x)
```

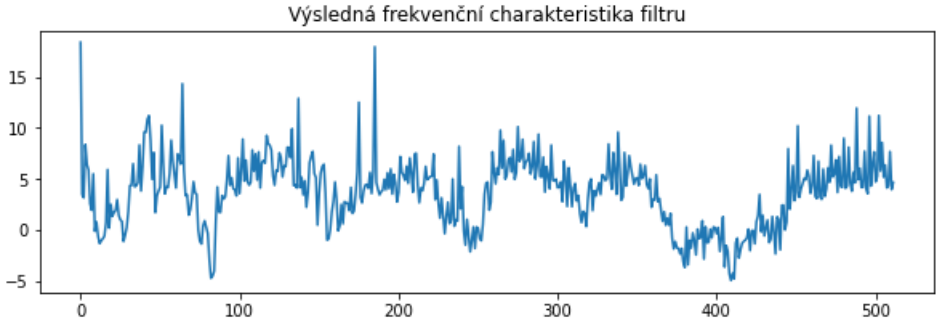
B)



Úkol 6

Bylo vytvořeno nové pole do kterého byly uloženy podíly signálu s rouškou děleného tím bez roušky, vše v absolutní hodnotě. Pak byly hodnoty pro rámce zprůměrovány a po aplikaci logaritmu jsme dostali výslednou charakteristiku.
A)
Vztah pro výpočet frekvenční charakteristiky je $H(e^{j\omega})$=|maskon/maskoff|

B)



C)
Filtr způsobuje v určitých časech mírné zeslabení signálů. V grafu především před hodnotou 100 a kolem hodnoty 400.

Úkol 7

Pro impulsivní odezvu byla použita zprůměrovaná hodnota z předchozího úkolu, která se aplikovala na v inverzní DFT. Ta je stejné jako samotná DFT, jen je změněn vzorec pro výpočet imaginárního čísla a celková suma je podělena velikostí pole (N)
A)

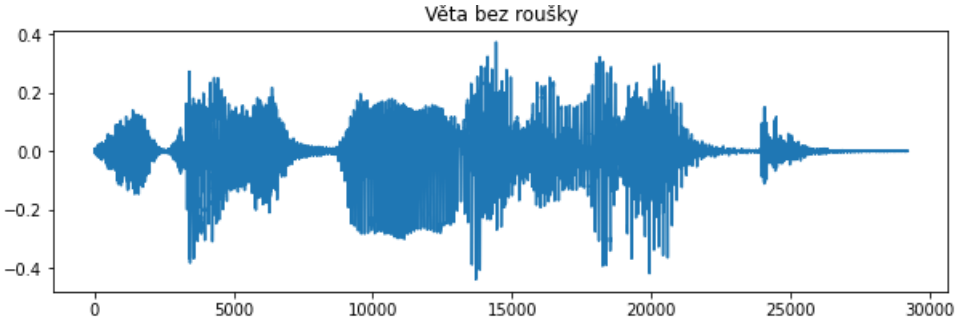
```
def idft(x):
    N=x.shape[0]
    n=np.arange(N)
    k=n.reshape((N,1))
    SUM=np.exp(2j*np.pi*k*n/N)
    return np.dot(SUM/N,x)
```

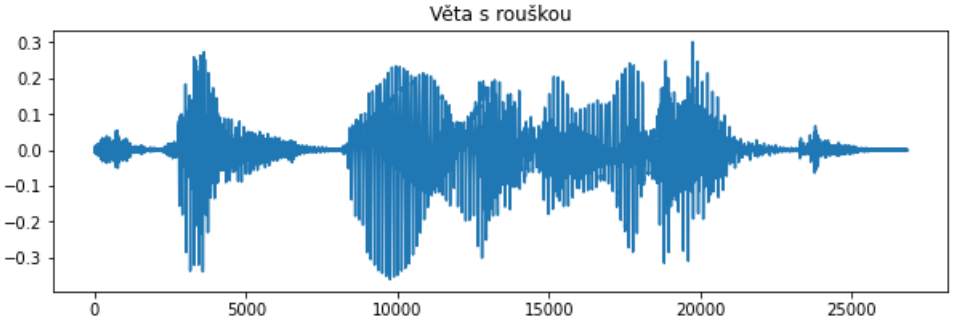
B)

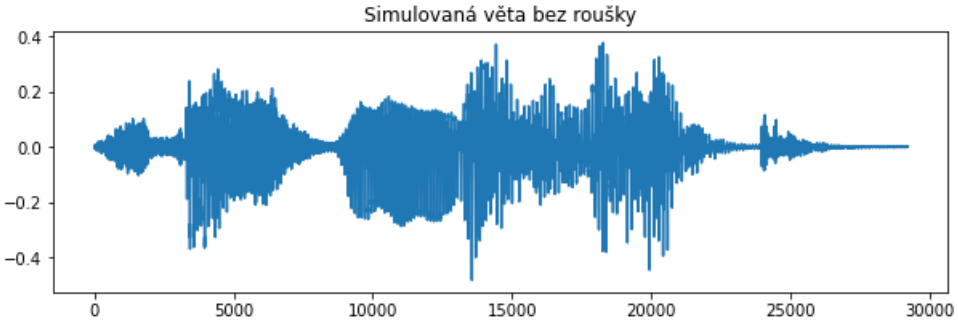
Úkol 8

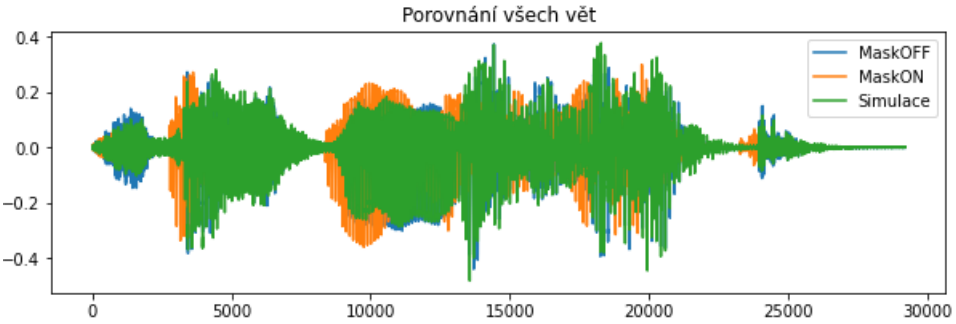
Na nahranou větu bez roušky byl pomocí scipy.signal.lfilter aplikován náš filtr, který by měl signál připodobnit nahrané větě s rouškou.

A)









B)

Signály po simulaci rouškou jsou docela podobné. Simulace proběhla dobře při hodnotách kolem 0, při hodnotách větších a menších již pak není simulace tak přesná.

Úkol 9 - Závěr

Myslím, že mé řešení v rámci možností funguje, především při zpracování tónu se mi výsledky zdají podařené. V následné aplikaci na větu už výsledky jsou odlišné, ale je možné že to je způsobeno nedokonalými nahrávkami. Zároveň šlo o mou první zkušenost s Pythonem, tudíž nějaké nepřesnosti mohou pramenit z né uplně ideálně naprogramovaných částí projektu.