ISS Protokol

Lukáš Baštýř (xbasty00)

January 4, 2021

Contents

1 Úkol 1&2 - tabulka velikostí

Název souboru	Délka	Vzorky
`maskon_tone.wav`	9.01s	1441508
`maskoff_tone.wav`	12.98s	207679
`maskon_sentence.wav`	5.76s	92240
`maskoff_sentence.wav`	6.71s	107360

Table 1: Tabulka velikosti souborů

Ještě mám v adresáři `/audio` 2 extra soubory. `maskoff_tone1s.wav` a `maskon_tone1s.wav`. Jendá se o manuálně ustřihlé vteřinové části (1s, 16000 vzorků), se kterými se dále pracuje. Místo řezání přímo v pythonu jsem to udělal mimo (audacity).

2 Úkol 3 - vzorec, graf

Vzorec pro výpočet velikosti rámce:

```
scope_lenght = 0.02 #délka rámce (20ms -> 0.02s)
time = 1            #délka audia (1s)
samples = 16031     #počet vzorků v 1 sekundě audia (soxi)
samples_per_scope = int(samples/(time/scope_lenght) #velikost rámce ve vzorcích)
```

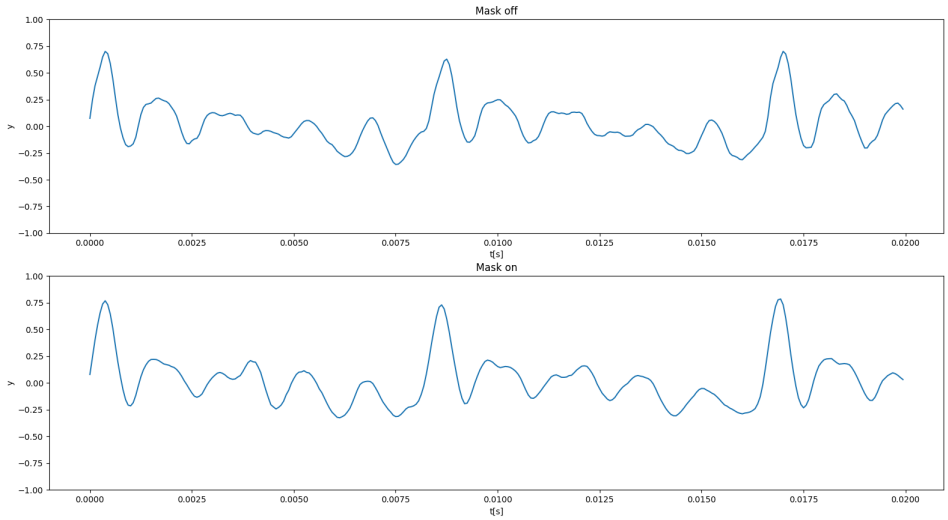

Figure 1: Graf rámce 0

3 Úkol 4 - Autokorelace, střední hodnota/rozptyl

Z grafu je krásně vidět, že opravdu nejsem zpěvák a nedokáži ani udržet jeden tón :/

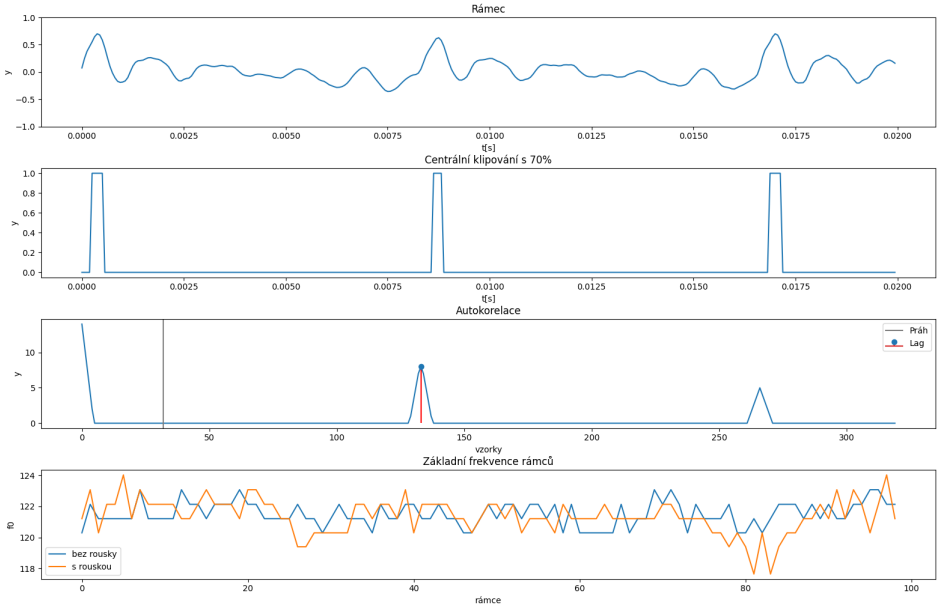

Figure 2: Grafy klipování, autokorelace a z. frekvence

Střední hodnota:

Bez masky:	121.52
S maskou:	121.38

Table 2: Střední hodnota

Rozptyl:

Bez masky:	0.5856
S maskou:	1.2373

Table 3: Rozptyl

Celková velikost změny by se dala změnt snížením vzorkovací frekvence, protože `f0` se získá dělením vzorkovací frekvence pozicí lagu.

4 Úkol 5 - DFT

DFT funkce:

```
def dft(signal, N):
    signal_copy = np.zeros(1024, dtype='complex_')
    signal_copy = np.pad(signal, (0, N - samples_per_scope))
    result = np.zeros(N, dtype='complex_')
    for k in range(0, N):
        sums = 0
        for n in range(0, N):
            sums += signal_copy[n] * cmath.exp( (-2j * np.pi * k * n) / N)
        result[k] = sums
    return result
```

Která je volána pro jednotlivé rámce následovně:

```
off_dft = np.zeros(shape=(frames, 512))
off_dft_padded = np.zeros(shape=(frames, 1024), dtype='complex_')

for i in range(0, frames):
    off_dft_padded[i] = dft(off_array[i], 1024)
    for k in range(0, 512):
        off_dft[i][k] = 10 * (np.log10(np.abs(off_dft_padded[i][k])**2))
```

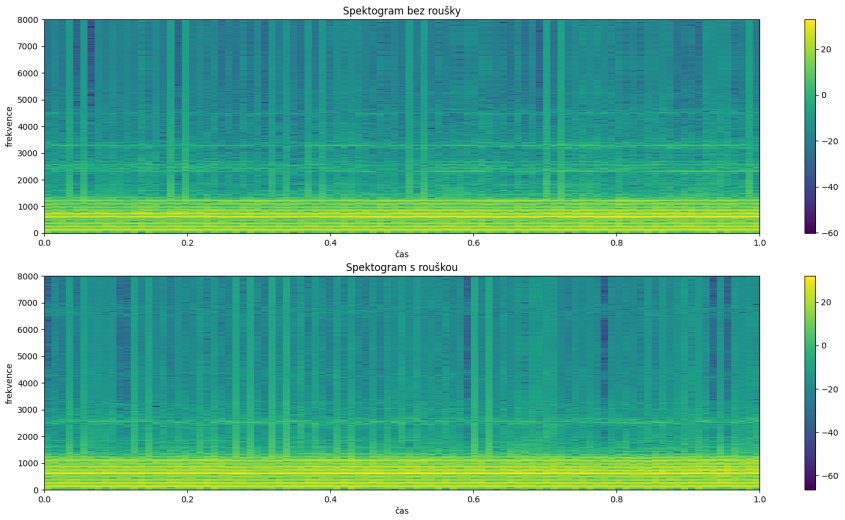

Figure 3: Spektogram bez a s rouškou

5 Úkol 6 - frekvenční charakteristika roušky

Vzorec pro výpočet H($e^{i\omega}$):

$$H(e^{i\omega}) = \frac{\sum_{n=0}^{N} B[n](e^{i\omega^{-n}})}{\sum_{m=0}^{M} A[m](e^{i\omega^{-m}})}$$

$$H(e^{i\omega}) = \frac{b[0] + b[1](e^{-i\omega}) + ... + b[n](e^{-i\omega})}{a[0] + a[1](e^{-i\omega}) + ... + a[m](e^{-i\omega})}$$

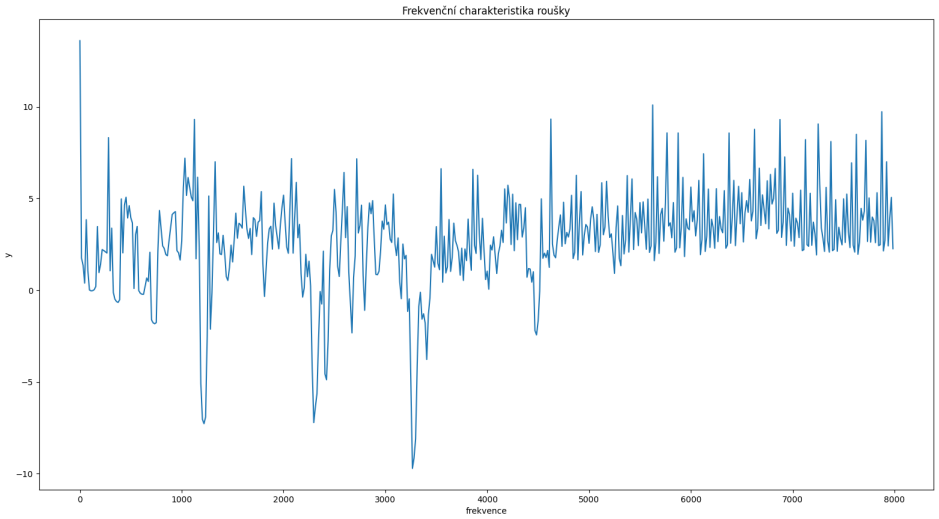

Figure 4: Graf frtekvenční charakteristiky roušky

Z grafu si dovoluju usoudit, že vyšší frekvence bude filtr zvětšovat a většinu nižších naopak zmenšovat.

6 Úkol 7 - IDFT

IDFT funkce:

```
def idft(signal, N):
    signal_copy = np.zeros(1024, dtype='complex_')
    signal_copy = np.pad(signal, (0, N - samples_per_scope))
    result = np.zeros(N, dtype='complex_')
    for k in range(0, N):
        sums = 0
        for n in range(0, N):
            sums += signal_copy[n] * cmath.exp( (2j * np.pi * k * n) / N)
        result[k] = (sums / N)
    return result
```

Která je volána následovně:

```
idft_res = idft(freq_char_prum, 1024)[:512]
```

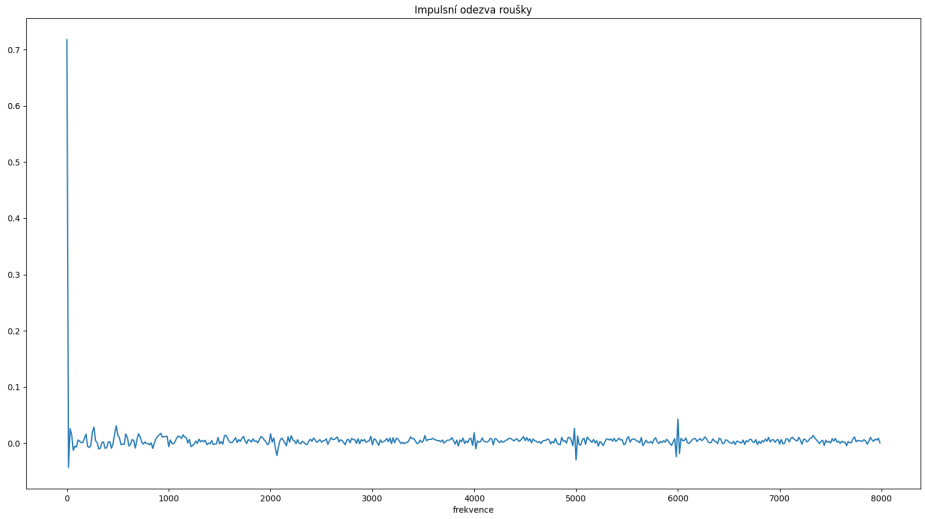

Figure 5: Graf impulsní odezvy roušky

7 Úkol 8 - Aplikace filtru

Co se týče rozdílů mezi signálem s rouškou a simulovanou rouškou, ty jsou celkem veliké. Hlavně je vidět, že věta s rouškou je trochu delší než věta bez roušky (to ale nemá na výslednou simulovanou roušky žádný vliv). Simulovaná rouška se mnohem více podobá signálu bez roušky než s rouškou. Signál s rouškou je mnohem více "utlumený/vyhlazený", což se prakticky neprojevilo na signálu se simulovanou rouškou. Nicméně je tam vidět "mix" obou signálu. Například ten vysoký "výkyv" kolem vzorku 9000. V simulovaném signálu je vidět kombinace jak signálu bez roušky, tak signálu s rouškou. Tudíž ten filtr očividně má na ten signál nějaký "rouškový" vliv.

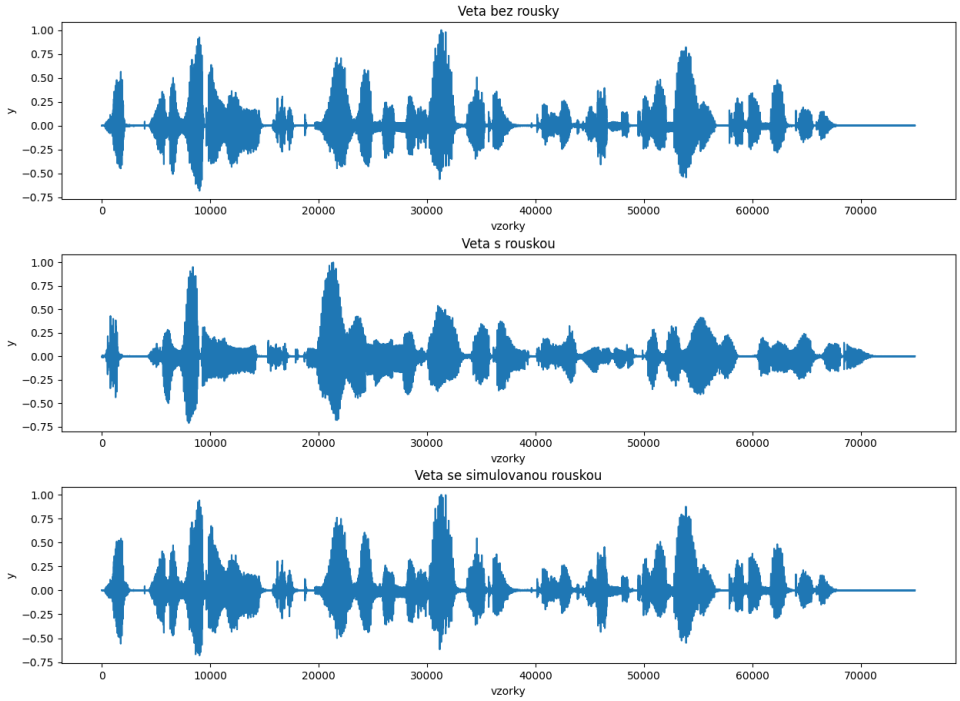

Figure 6: Graf celého signálu

8 Závěr

Přesně jak je napsáno v zadání, mé výsledky jsou opravdu "nic moc". Spíše než rouškový filtr to signál pouze zesílilo. Ale, nějaká změna tam slyšet je. Možná to se dá považovat za rouško. Ale z poslechu bych řekl, že moje řešení asi příliš nefunguje. Jednotlivé části mi přijdou funkční (například vlastní DFT, IDFT, autokorelace), takže je někde asi přímo chyba v implementaci.

Určitě se nejednalo o jednoduchý projekt, ale byl to zatím asi ten nejzajímavější. Člověk to mohl řešit jak chtěl, kdy chtěl a prakticky co chtěl. Za mě asi jeden z nejzajímavějších projektů. Akorát je teda škoda, že výsledek se tak moc liší od představy (signál s rouškou).

9 Extra úkol 12 - n-násobný lag

Vzniklá chyba se dá odstranit získáním mediánu ze všech rámců. Následně kontroloval, jestli se nějaký rámec hodně liší od ostatních. Pokud ano, nahradí se hodnota toho rámce mediánem.

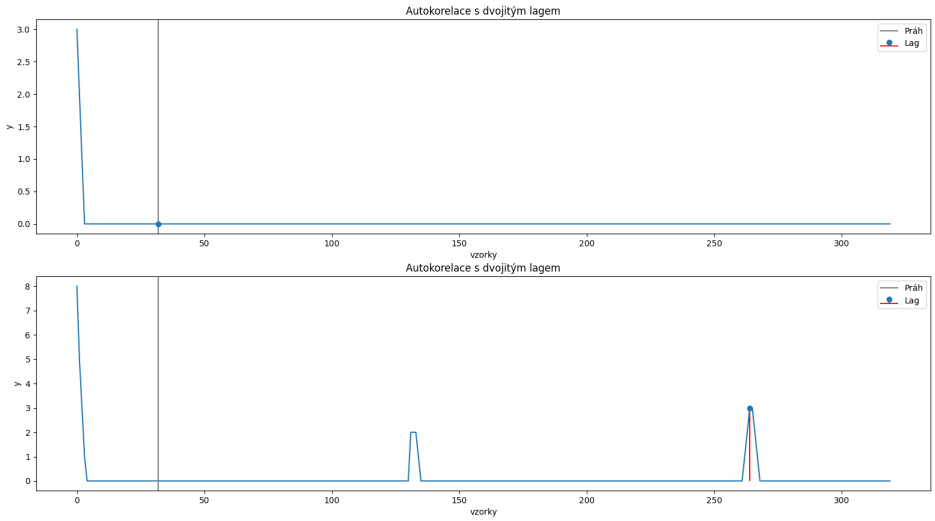

Figure 7: Grafy špatných lagů

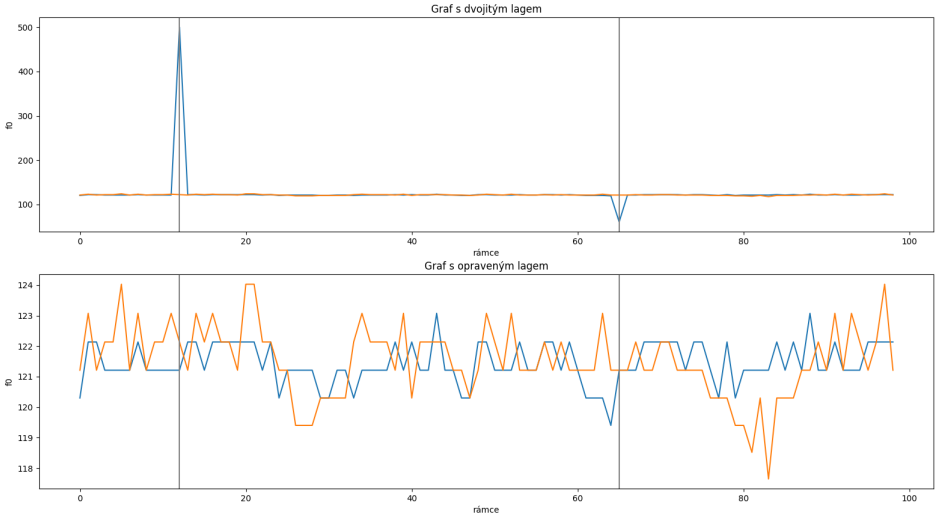

Figure 8: Grafy f0 špatného a opraveného lagu

10 Extra úkol 15 - Fázový posun

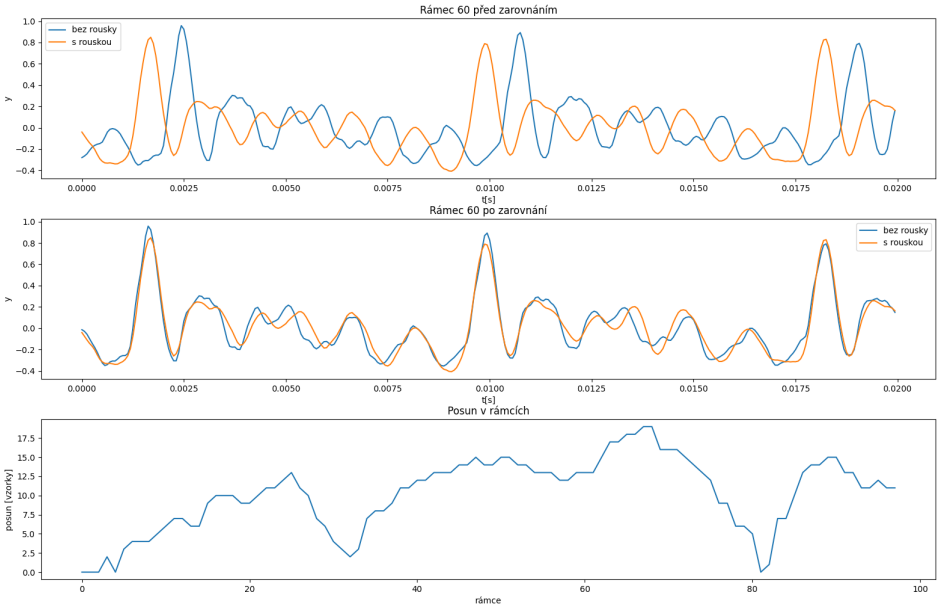

Figure 9: Grafy posunu