Adrián Bobola – xbobol00

ISS – Projekt 2020/21 Fakulta informačních technologií – VUT v Brně

Úloha 1:

Názov súboru	Dĺžka nahrávky [vzorky]	Dĺžka nahrávky [s]
maskoff_tone.wav	117 077	7,32
maskon_tone.wav	136 875	8,55

Vstupné signály načítame pomocou príkazu wavfile.read z knižnice wavfile.

Úloha 2:

Názov súboru	Dĺžka nahrávky [vzorky]	Dĺžka nahrávky [s]
maskoff_sentence.wav	112 299	7,02
maskon_sentence.wav	105 131	6,57

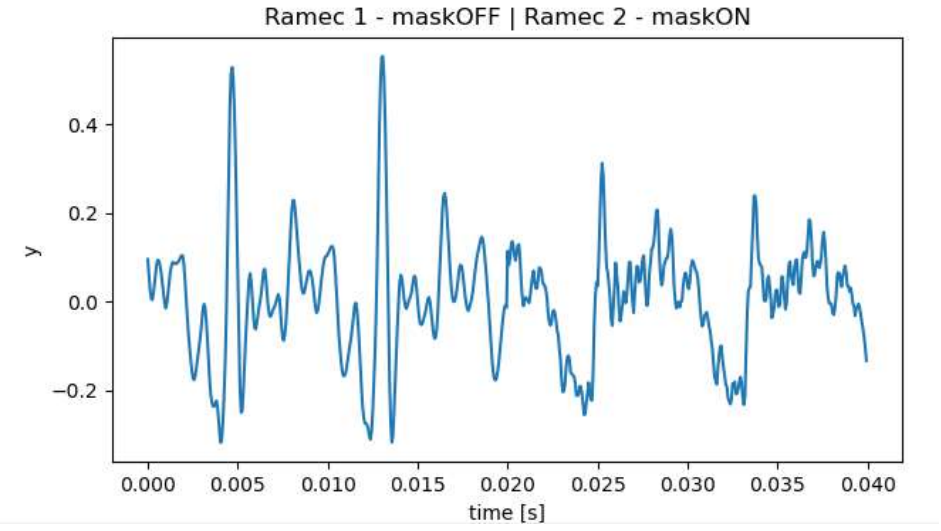
Úloha 3:

Signály _tone extrahujeme o dĺžke 1.01 sekúnd (pre 100 rámcov).

Následne signály ustredníme a normalizujeme do dynamického rozsahu [-1;1].

Pri vzorkovacej frekvencii F_s = 16 kHz bude mať 1 ms nahrávky 16 vzoriek. Keďže sme použili 20ms/rámec (zo zadania), 1 rámec bude mať v našom prípade 20 x 16 = 320 vzoriek. Platí teda vzťah:

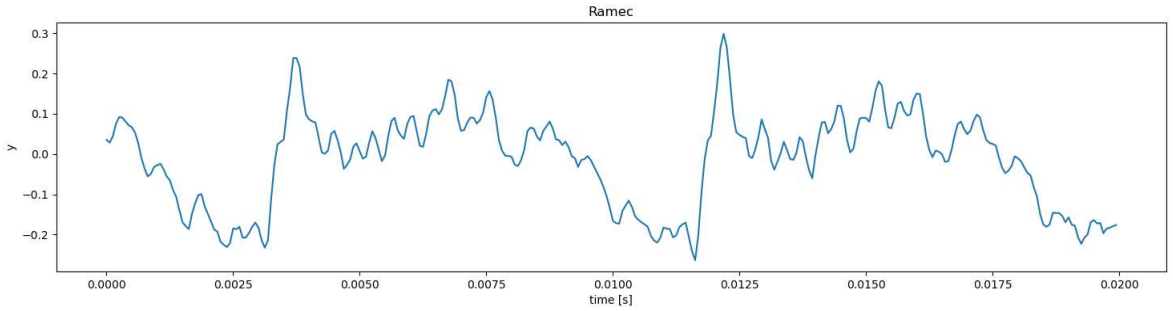
$$Velkosť\ rámca[počet\ vzoriek] = Fs[Hz]\ x\ dĺžka\ rámca[ms]$$



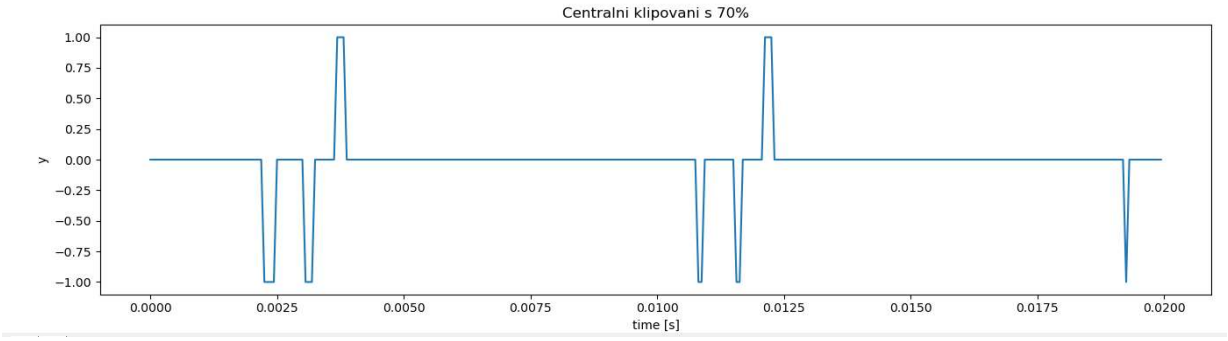
Obr. 1. Graf dvoch zvolených rámcov – jeden s rúškom, druhý bez rúška

Úloha 4:

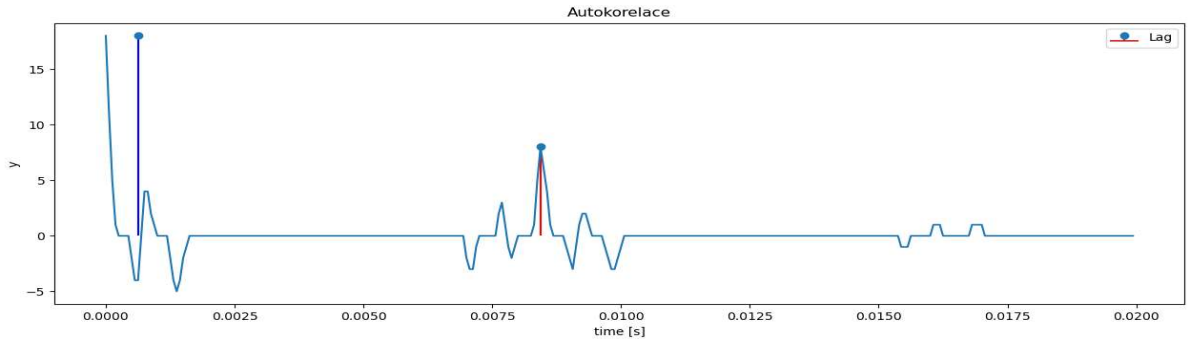
Pre získanie základného tónu f_0 aplikujeme Centrálne klipovanie s 70% a následne Autokoreláciu.



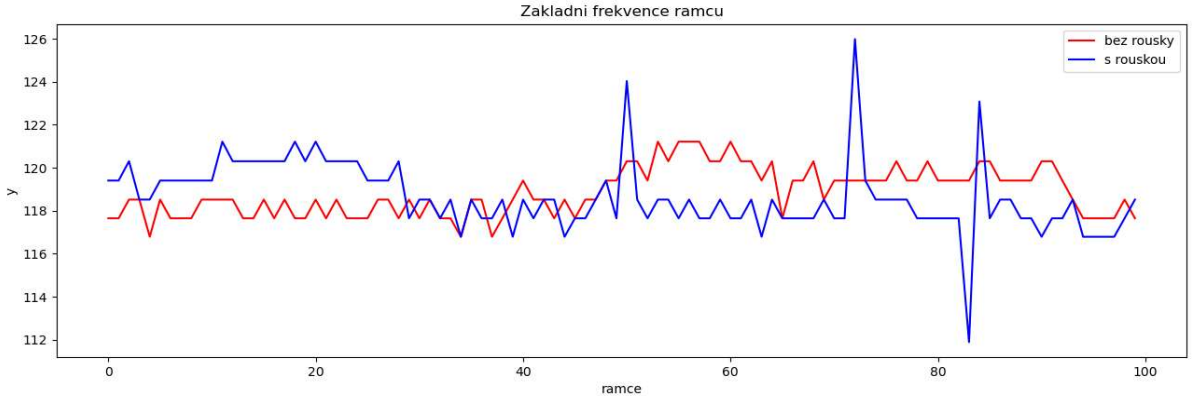
Obr. 2. Zvolený ľubovoľný rámec



Obr. 3. Aplikované centrálne klipovanie s 70%



Obr. 4. Aplikácia autokorelácie



b) Stredná hodnota a rozptyl základnej frekvencie:

Nahrávka	Stredná hodnota	Rozptyl
maskon_tone.wav	118,846	2,735
maskoff_tone.wav	118,629	1,215

Určíme strednú hodnotu pomocou funkcie numpy.mean a rozptyl pomocou funkcie numpy.var

c) Keďže frekvenciu f_0 počítame ako F_s[Hz]/ hodnota „lag-u“ – v našom prípade 16 000 / lag, aj drobné zmeny „lag-u“ majú celkom veľký vplyv na výslednú frekvenciu.

Výrazné zmeny f_0 by sa dali zmenšiť napr. spriemerovaním hodnôt „lag-u“ pre n-rámcov napr. hodnotu „lag-u“ pre rámec č. 1 by sme vypočítali ako priemer hodnôt „lag-u“ pre rámec č. 0 a 1. Prípadne pre viac rámcov. Tým by sme minimalizovali výrazne výkyvy frekvencie.

Úloha 5:

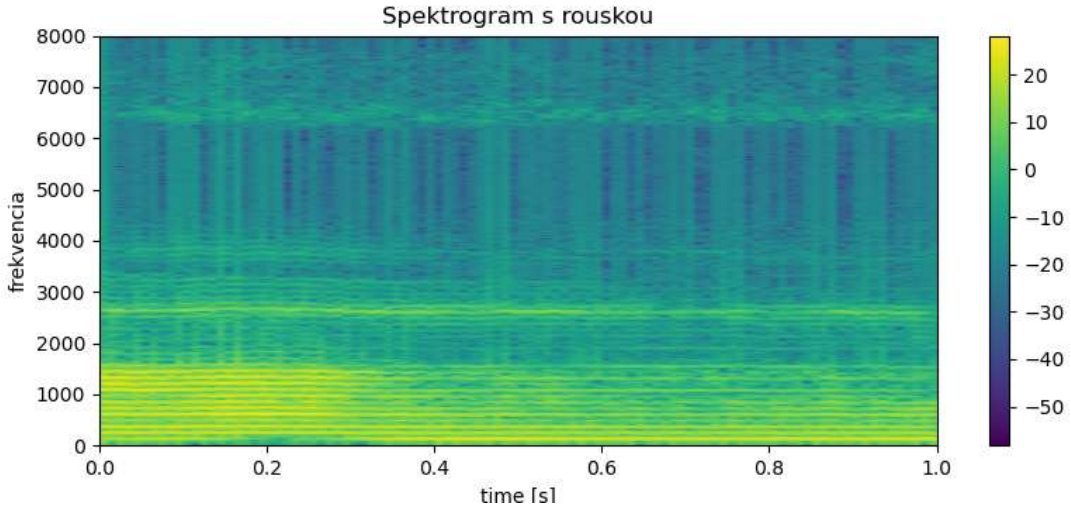
a) Funkcia implementujúca DFT:

```
def vlastne_DFT(ramec_vybrany):
    dft_vysledok = []
    pi = cmath.pi
    for k in range(len(ramec_vybrany)):
        dft_helper = 0
        for n in range(len(ramec_vybrany)):
            exponent = cmath.exp(-2j * pi * (1 /
len(ramec_vybrany)) * k * n)
            dft_helper += (ramec_vybrany[n] * exponent)
        dft_vysledok.append(dft_helper)
    return dft_vysledok
```

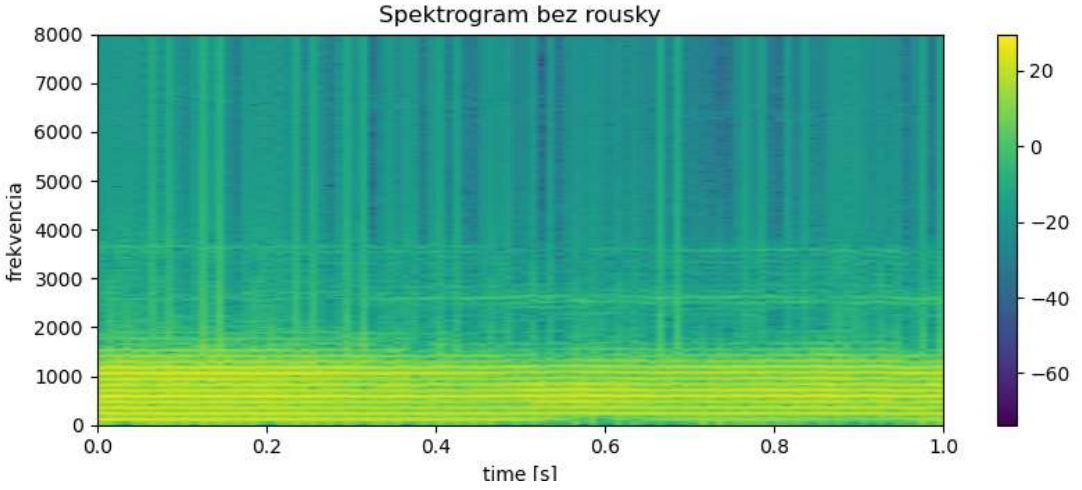
Vstup: rámec nad ktorým chceme spraviť DFT

Výstup: pole vypočítaných DFT hodnôt - pre každú vzorku rámca

b) Spektrogram pre tón s rúškom:



Spektrogram pre tón bez rúška:

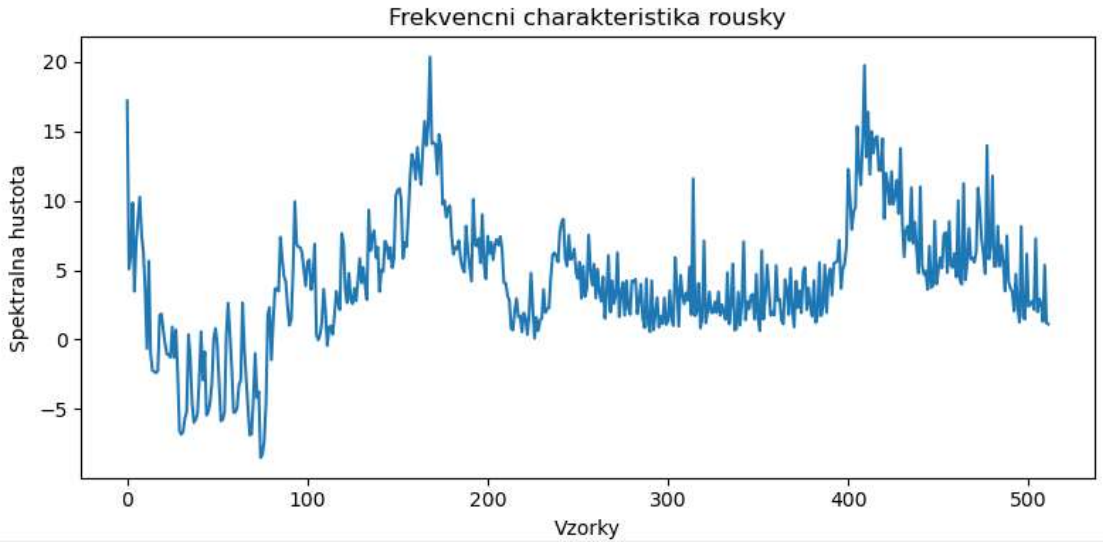


Úloha 6:

a) $H\left(e^{j\omega}\right) = \frac{Y(e^{j\omega})}{X(e^{j\omega})}$

Kde Y sú konkrétne rámce s aplikovanou DFT z nahrávky maskon_tone a X reprezentuje rámce s DFT z nahrávky maskoff_tone

b) Frekvenčná charakteristika rúška:



c) Filter sa podobá na lineárny filter - pásmová zadrž.
Filter tlmí určité frekvencie zo vstupného signálu.

Úloha 7:

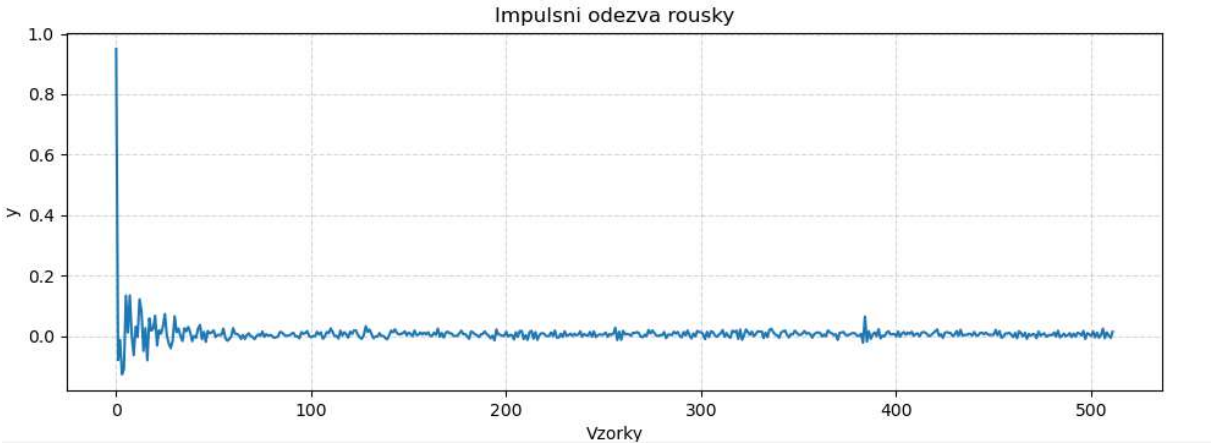
a) Vlastná implementácia IDFT:

```
def vlastne_IDFT(ramec_vybrany):
    idft_vysledok = []
    pi = cmath.pi
    for k in range(len(ramec_vybrany)):
        idft_helper = 0
        for n in range(len(ramec_vybrany)):
            exponent = cmath.exp(2j * pi * (1 /
len(ramec_vybrany)) * k * n)
            idft_helper += (ramec_vybrany[n] * exponent)
        idft_vysledok.append(idft_helper * (1 /
len(ramec_vybrany)))
    return idft_vysledok
```

Vstup: rámec nad ktorým chceme spraviť IDFT

Výstup: pole vypočítaných IDFT hodnôt - pre každú vzorku rámca

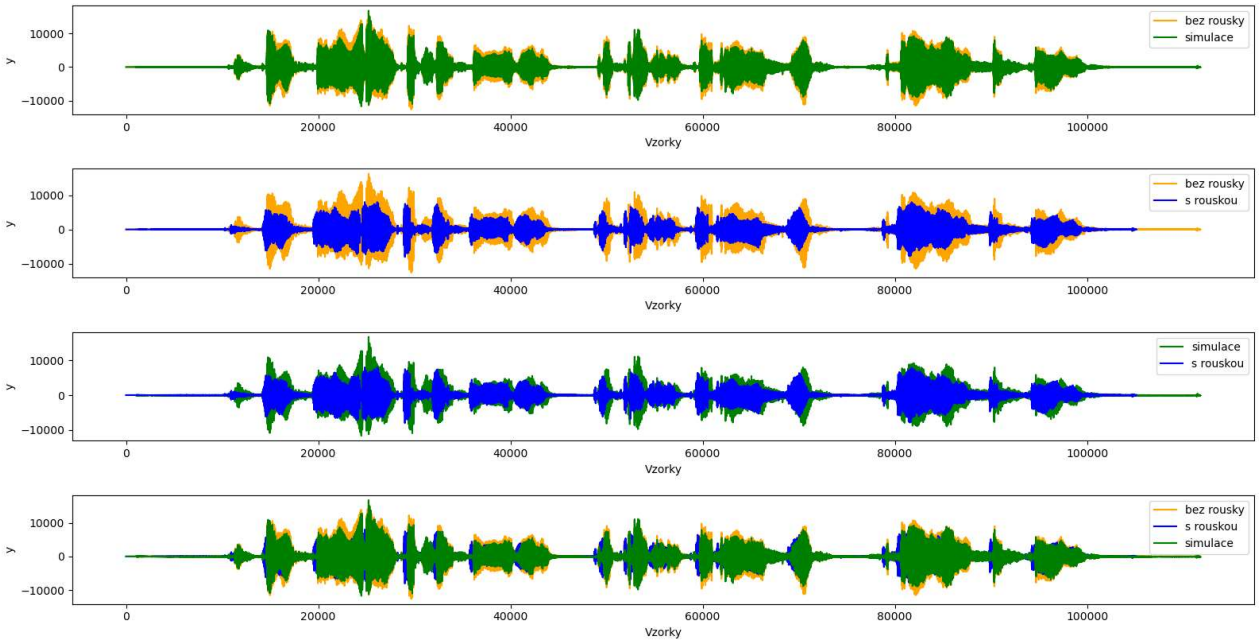
b) Graf impulznej odozvy rúška:



Úloha 8:

a) Grafy:

- 1. graf - maskoff_sentence, sim_maskon
- 2. graf - maskoff_sentence, maskon_sentence
- 3. graf - maskon_sentence, sim_maskon
- 4. graf - maskoff_sentence, maskon_sentence, sim_maskon



Signál z vety s rúškom sa od simulovaného signálu mierne líši, ale aj napriek tomu si dovolím tvrdiť že signály sú takmer podobné.

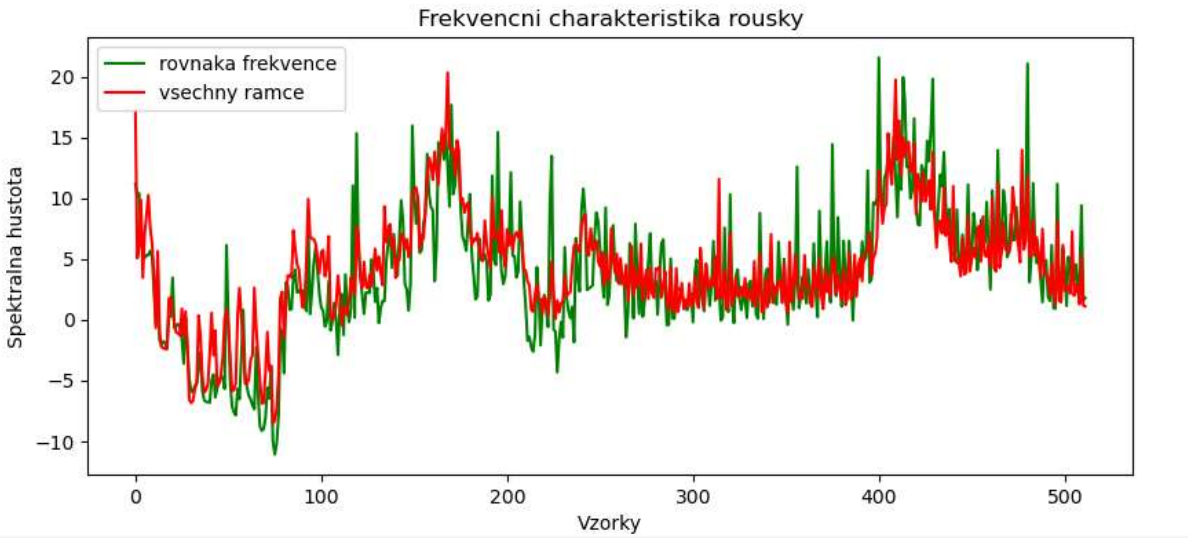
Najčastejšie nepresnosti mi prídu pri vyšších tónoch nahrávky bez rúška, kde je simulovaný signál o niečo vyšší ako originál signál vety bez rúška.

Úloha 9 – Záver:

Implementované riešenie projektu funguje. Pri testovaní som nezaznamenal žiadne problémy alebo výrazne odchýlky medzi signálmi.

Najväčší problém pri riešení projektu som mal práve pri vytváraní nahrávok, kde sa mi nedarilo vysloviť samohlásku „á“ s rovnakou výškou hlasu pri nahrávke s rúškom a bez rúška. Bohužiaľ ani použitá aplikácia ladičky mi veľmi nepomohla – hlasivkám nerozkážem. Mierne výkyvy výšok tónu medzi nahrávkami spôsobili drobné nepresnosti pri vytváraní filtra.

Úloha 13:



Po aplikácii tejto metódy sme dostali o niečo presnejšiu charakteristiku rúška, pretože sme vyberali iba rámce z nahrávky s rovnakou základnou frekvenciou. Výkyvy frekvencie v nahrávkach tak nehrali významnú úlohu pri charakteristike.