Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Percepcja maszyn

Sprawozdanie z laboratorium L4+L5

Kaniuka Jan

Warszawa, 2022

Spis treści

1. Filtracja sygnału audio

1.1. Treść zadania

Na dołączonym do zadania nagraniu, oprócz rozmowy dwóch osób, pojawiają się dwa rodzaje zakłóceń. Pierwszym jest **stały przydźwięk spowodowany sprzężeniem sieciowym**. Drugie zakłócenie pojawia się pod koniec nagrania - jest to **sygnał budzika**. Proszę przefiltrować nagranie w taki sposób, aby oba te zakłócenia usunąć, pozostawiając przy tym możliwie najmniejsze zniekształcenia właściwej mowy.

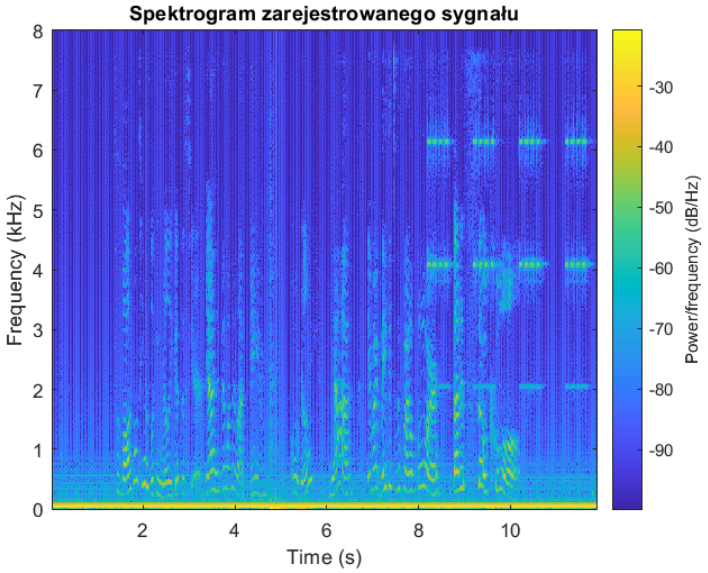
W ramach zadania proszę przygotować zestaw filtrów typu *windowed-sinc*, które będą wycinały odpowiednie pasma częstotliwości. Analizę pasm do wycięcia proszę wykonać "naocznie" na podstawie analizy spektrogramu nagrania. Filtry powinny być wyznaczane zgodnie z metodami przestawionymi na wykładzie. W ramach rozwiązania proszę przygotować co najmniej dwa różne filtry: filtr dolnoprzepustowy oraz pasmowoblokujący.

Filtracji sygnału proszę dokonać przez wykorzystanie standardowej funkcji splotu (`conv`). Proszę pamiętać o tym, że parametry filtrów wyznaczane są w relacji do częstotliwości próbkowania, a więc częstotliwości wyrażone w *Hz* należy odpowiednio przeliczać na ułamek [0 - 0.5].

1.2. Rozwiązanie

1.2.1. Analiza spektrogramu surowego sygnału

Rozwiązywanie zadania zacząłem od analizy spektrogramu otrzymanego sygnału audio. Wykorzystałem w tym celu funkcję `spectrogram` pakietu `MATLAB`.



Rys. 1.1. Spektrogram sygnału wejściowego

1.2.2. Zidentyfikowanie zakresów częstotliwości do wycięcia

Należy oczyścić sygnał z dwóch rodzajów zakłóceń. Są to odpowiednio:

— **przydźwięk spowodowany sprzężeniem sieciowym** - jest on słyszalny przez cały czas trwania nagrania. Z teoretycznego punktu widzenia jest to zwyczajnie słyszalne nałożenie na sygnał audio częstotliwości sieci zasilającej AC 50Hz (składowa podstawowa) oraz jej harmonicznych (100Hz, 150Hz, itd.). Na spektrogramie jest on widoczny jako pasmo o żółtym kolorze w zakresie niskich częstotliwości.

Do usunięcia przydźwięku planuję zastosować **filtr górnoprzepustowy** o częstotliwości granicznej 200Hz i szerokości pasma przejściowego 100Hz (pasmo przejściowe nie powinno być ekstremalnie wąskie, ponieważ wtedy nagranie po obróbce będzie brzmiało *nienaturalnie*).

— **sygnał budzika** - jest słyszalny jako okresowe pikanie, niezbyt przyjemne dla ucha (szczególnie o poranku). Sygnał pojawia się około ósmej sekundy nagrania. Na spektrogramie widać wyraźnie, że sygnał budzika jest złożony z trzech głównych składowych częstotliwościowych o częstotliwościach: 2kHz, 4kHz i około 6kHz.

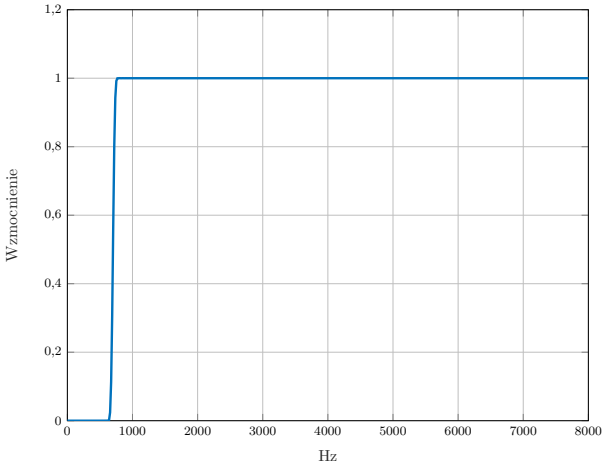
Do usunięcia sygnału budzika planuję zastosować **3 filtry pasmowozaporowe** blokujące częstotliwości z zakresu o środku w częstotliwościach odpowiednio 2kHz, 4kHz i 6kHz i o szerokości pasma przejściowego 150Hz. Szerokość pasma przejściowego wybrałem zgodnie z zasadą, że im wyższe pasmo chcemy wytłumić, tym szersze powinno być pasmo przejściowe (aby zachować w miarę naturalne brzmienie). Szerokość pasm zaporowych wybiorę na drodze eksperymentu podczas testowania filtrów.

1.2.3. Charakterystyka amplitudowa przygotowanych filtrów

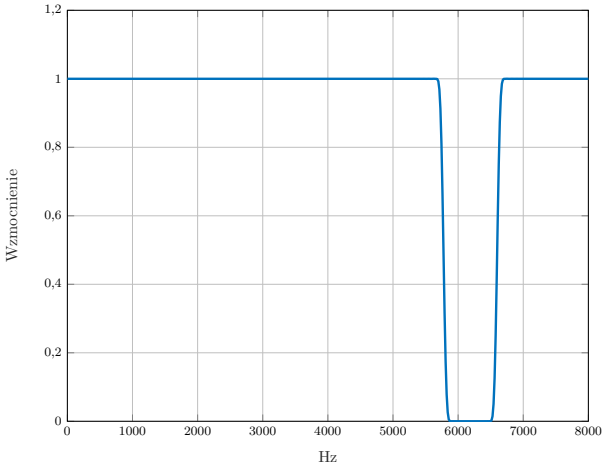
Przygotowałem w sumie cztery filtry typu *windowed-sinc* (chociaż wykorzystam prawdopodobnie jedynie dwa z nich):

— filtr dolnoprzepustowy → *lowpass(sig, fc, BW)*
— filtr górnoprzepustowy → *highpass(sig, fc, BW)*
— filtr pasmowoprzepustowy → *bandpass(sig, fL, fH, BW)*
— filtr pasmowozaporowy → *bandstop(sig, fL, fH, BW)*

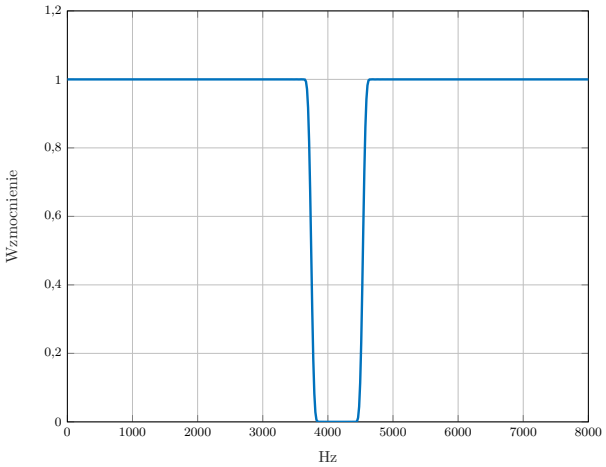
Poniżej zamieściłem charakterystyki amplitudowe przygotowanych filtrów (wzmocnienie w dziedzinie częstotliwości). Wykresy wygenerowałem używając zmodyfikowanej funkcji `filtstat()` z laboratorium trzeciego.



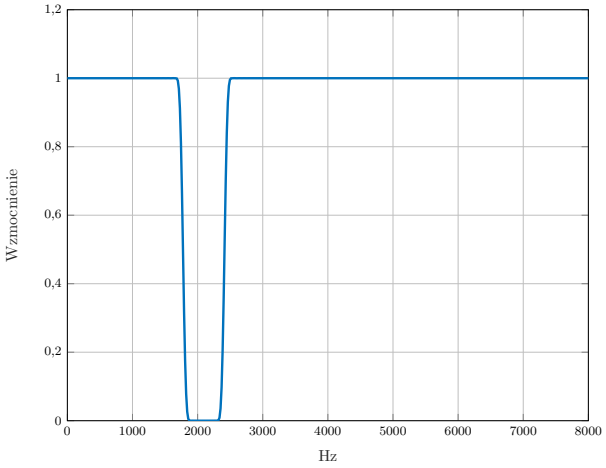
Rys. 1.2. Filtr highpass, fc=700Hz



Rys. 1.3. Filtr bandstop #1, fL=5775Hz , fH=6600Hz



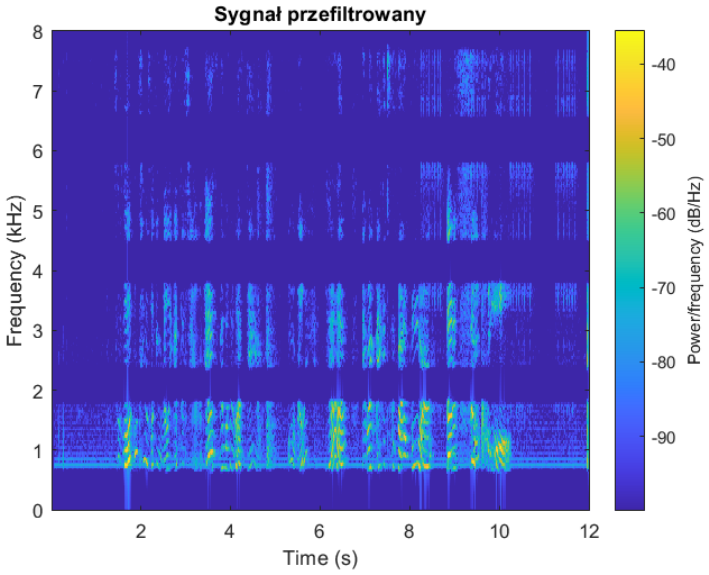
Rys. 1.4. Filtr bandstop #2, fL=3744Hz , fH=4538Hz



Rys. 1.5. Filtr bandstop #3, fL=1775Hz , fH=2413Hz

1.2.4. Analiza spektrogramu przefiltrowanego sygnału

Po eksperymentalnym procesie doboru szerokości pasm zaporowych w filtrach pasmowozaporowych udało się osiągnąć znaczną poprawę jakości nagrania. Zwiększyłem też częstotliwość graniczną filtru górnoprzepustowego do 700Hz - był to **kompromis pomiędzy dostatecznym wytłumieniem sprzężenia sieciowego a możliwie niewielkim zniekształceniem mowy**. W celu sprawdzenia, czy filtry *bandstop* faktycznie blokują zadane pasmo częstotliwości, wygenerowałem spektrogram nagrania audio po edycji. Rezultat widoczny jest na poniższym wykresie.



Rys. 1.6. Spektrogram sygnału po filtracji

1.2.5. Ocena jakości otrzymanego nagrania

Ważniejsza od spektrogramu jest nauszna ocena jakości nagrania. Uważam, że udało się osiągnąć wysoce zadowalającą jakość nagrania audio. Filtr górnoprzepustowy pozwolił na wyeliminowanie przydźwięku do poziomu lekkiego szumienia w tle, które absolutnie nie zagłusza osób rozmawiających na nagraniu. Filtry pasmowozaporowe wyeliminowały składowe częstotliwościowe sygnału budzika. Nie słychać już głośnego, piskliwego i rozpraszającego *pikania*. Zamiast tego

w tle słychać jedynie ciche, okresowe stukanie, które jest pozostałością po filtracji sygnału. Jest ono jednak na tyle wytłumione, że bez trudu można zrozumieć treść rozmowy. Próbowałem na różne sposoby wyeliminować ten stukot poprzez np. zwiększenie pasm zaporowych filtrów *bandstop*. Niestety bardziej zniekształcało to dźwięki rozmowy aniżeli sam stukot.

W tym miejscu można odpowiedzieć sobie na pytanie, czy wspomniany *"stukot"* należy wyeliminować całkowicie. Osobiście uważam, że nie jest to konieczne. Jeżeli przykładowo analityk wywiadu otrzymałby takie nagranie to filtrowałby je do momentu, aż jego treść (np. rozmowa) będzie po prostu zrozumiała, a nie idealna i całkowicie pozbawiona zakłóceń.

1.2.6. Kody źródłowe

Listing 1.1. Filtr lowpass

```
function [y,kernel] = lowpass(sig, fc, BW)

M=floor(4/BW); % długość jądra
if mod(M,2)==1 % M musi być parzyste
    M=M+1;
end

i=0:M;
sinc_func=2*pi*fc*sinc(2*fc*(i-M/2)); % funkcja sinc
window=0.42-0.5*cos(2*pi*i/M)+0.08*cos(4*pi*i/M); % okno Blackmana
kernel=(sinc_func.*window);

% zapewnienie wzmocnienia=1 dla zerowej częstotliwości
kernel=kernel/sum(kernel);

y=conv(sig,kernel); % splot sygnału z jądrem filtru

end
```

Listing 1.2. Filtr highpass

```
function [y,kernel] = highpass(sig, fc, BW)

M=floor(4/BW); % długość jądra
if mod(M,2)==1 % M musi być parzyste
    M=M+1;
end

i=0:M;
sinc_func=2*pi*fc*sinc(2*fc*(i-M/2)); % funkcja sinc
window=0.42-0.5*cos(2*pi*i/M)+0.08*cos(4*pi*i/M); % okno Blackmana
kernel=(sinc_func.*window);

% zapewnienie wzmocnienia=1 dla zerowej częstotliwości
kernel=kernel/sum(kernel);

% inwersja spektralna
kernel=-kernel; %zmiana znaku wszystkich próbek
kernel(kernel==min(kernel))=kernel(kernel==min(kernel))+1;

y=conv(sig,kernel); % splot sygnału z jądrem filtru

end
```

Listing 1.3. Filtr bandpass

```
function [y,kernel] = bandpass(sig, fL, fH, BW)

M=floor(4/BW); % długość jądra
if mod(M,2)==1 % M musi być parzyste
    M=M+1;
end

i=0:M;
sinc_func_LOW=2*pi*fL*sinc(2*fL*(i-M/2)); % lowpass 1
sinc_func_HIGH=2*pi*fH*sinc(2*fH*(i-M/2)); % lowpass 2

window=0.42-0.5*cos(2*pi*i/M)+0.08*cos(4*pi*i/M); % okno Blackmana

fL=sinc_func_LOW.*window;
fH=sinc_func_HIGH.*window;

% zapewnienie wzmocnienia=1 dla f=0
fL=fL/sum(fL); % filtr LP1
fH=fH/sum(fH); % filtr LP2

% filtr lowpass2 poddajemy inwersji spektralnej
fH=-fH;
fH(fH==min(fH))=fH(fH==min(fH))+1;

% złożenie jąder filtrów LP i HP w jedno
kernel=(fL+fH);

% ponowna inwersja w celu otrzymania filtra bandpass z filtra bandstop
kernel=-kernel;
kernel(kernel==min(kernel))=kernel(kernel==min(kernel))+1;

y=conv(sig,kernel); % splot sygnału i jądra filtru

end
```

Listing 1.4. Filtr bandstop

```
function [y,kernel] = bandstop(sig, fL, fH, BW)

M=floor(4/BW); % długość jądra
if mod(M,2)==1 % M musi być parzyste
    M=M+1;
end

i=0:M;
sinc_func_LOW=2*pi*fL*sinc(2*fL*(i-M/2)); % lowpass 1
sinc_func_HIGH=2*pi*fH*sinc(2*fH*(i-M/2)); % lowpass 2

window=0.42-0.5*cos(2*pi*i/M)+0.08*cos(4*pi*i/M); % okno Blackmana

fL=sinc_func_LOW.*window;
fH=sinc_func_HIGH.*window;

% zapewnienie wzmocnienia=1 dla f=0
fL=fL/sum(fL); % filtr LP1
fH=fH/sum(fH); % filtr LP2

% filtr lowpass2 poddajemy inwersji spektralnej
fH=-fH;
fH(fH==min(fH))=fH(fH==min(fH))+1;

% złożenie jąder filtrów LP i HP w jedno
```

```
kernel=(fL+fH);

y=conv(sig,kernel); % splot sygnału i jądra filtru

end
```

Listing 1.5. Aplikacja filtrów do sygnału

```
clear all;

[sig_in, fs] = audioread('noised.wav');

[highpass_out, kernel_HP]=highpass(sig_in(:,1), 700/fs, 100/fs);
[BS_1_out, kern_BS1]=bandstop(highpass_out, 5775/fs, 6600/fs, 150/fs);
[BS_2_out, kern_BS2]=bandstop(BS_1_out, 3744/fs, 4538/fs, 150/fs);
[BS_3_out, kern_BS3]=bandstop(BS_2_out, 1775/fs, 2413/fs, 150/fs);

filtered_signal = BS_3_out;
sound(filtered_signal, fs);

figure;
win_len = 512;
win_overlap =256;
nfft = 512;
spectrogram(BS_3_out, win_len, win_overlap, nfft, fs, ...
    'MinThreshold', -100, 'yaxis');
title(sprintf('Sygnał przefiltrowany'));
```

Listing 1.6. Spektrogram i odsłuchanie nagrania

```
[sig, fs] = audioread('noised.wav');
win_len = 512;
win_overlap =256;
nfft = 512;

figure;
spectrogram(sig(:,1), win_len, win_overlap, nfft, fs, ...
    'MinThreshold', -100, 'yaxis');
title("Spektrogram zarejestrowanego sygnału");

sound(sig,fs);
```

Listing 1.7. Rysowanie charakterystyki amplitudowej

```
function filtstat(f)
    nfft = 1000;
    fs = 1000;

    f_ex = zeros(1, nfft);
    f_ex(1:size(f, 2)) = f;

    y=fft(f_ex);
    f_base = linspace(0, fs/2, nfft/2+1);
    amp = abs(y(1:nfft/2+1));
    plot(16*f_base, amp, 'LineWidth',1.5);
    grid on;
    xlabel('Hz');
    ylabel('Wzmocnienie');
end
```