

Na czerwono oznaczono komentarze, bądź też elementy do usunięcia (ze względu, że np. pochodzą ze źródeł zewnętrznych a służą jedynie jako placeholder)

1. Temat

„Projekt i implementacja adaptacyjnego systemu korekcji parametrów nagrań dźwiękowych”

Celem pracy jest zaprojektowanie i implementacja adaptacyjnego systemu korekcji parametrów nagrań dźwiękowych z wykorzystaniem metod uczenia maszynowego.

1.1. Zadania:

1.1.a. Przygotowanie szkicu teoretycznego i raportu literaturowego

Literatura:

- Everest F. – Podręcznik akustyki
- Deep Room Recognition Using Inaudible Echos [DOI: 10.1145/3264945]

1.1.b. Projekt aplikacji

Nakładanie pogłosu

Zdejmowanie pogłosu

Aplikacja ma wyznaczać:

- Barwę dźwięku – ulepszona spectral centroid
- Klarowność – energia pierwszych 50ms vs reszta
- Czas pogłosu – T60/T30/Early Decay time
- Pierwsze odbicie

1.1.c. Przygotowanie środowiska pracy (instalacja oprogramowania i dodatkowych bibliotek programistycznych)

Python

1.1.d. Implementacja kodu

Pycharm

1.1.e. Testy aplikacji

Testy subiektywne oprogramowania – ocena działania programu w porównaniu z nagraniami wzorcowymi – ocena wierności odtworzenia charakteru nagrania, ocena naturalności brzmienia.

SPIS TREŚCI

1. TEMAT	1
1.1. ZADANIA:	2
2. WSTĘP	4
3. TEORIA	5
3.1. PODSTAWOWE ZAGADNIENIA PORUSZANE W PRACY	5
3.2. CZAS POGŁOSU	BŁĄD! NIE ZDEFINIOWANO ZAKŁADKI.
3.3. CHARAKTERYSTYKA CZĘSTOTLIWOŚCIOWA W ZALEŻNOŚCI OD DŁUGOŚCI IMPULSU.	7
3.4. BARWA DŹWIĘKU	12
4. PROJEKT APLIKACJI	13
4.1. MOŻLIWOŚCI	13
4.2. SCHEMAT DZIAŁANIA	15
4.3. IMPLEMENTACJA	16
4.4. POGŁOS	26
4.5. RÓWNOLEGŁOŚĆ PARAMETRYZOWANIA	26
5. PODSUMOWANIE	30
6. UCZENIE MASZYNOWE	29
6.1. WSTĘP	29

2. Wstęp

Celem pracy dyplomowej jest projekt oraz implementacja systemu analizy akustyki pomieszczenia, jako próbki nagrania audio w celu ekstrakcji parametrów akustycznych charakterystycznych dla tego nagrania, takich jak wpływ pomieszczenia na barwę dźwięku, czas pogłosu, czy też analiza odbić fali dźwiękowej. Parametry te nadają nagraniu charakterystyczne brzmienie.

Projektowany system dokonuje korekty nagrania, rozumianej jako manipulacja badanymi wcześniej parametrami dopasowując je według danych założeń do wzorca. Badana jest również zmiana tychże parametrów, która może się okazać kluczowa do późniejszego rozwoju programu o uczenie maszynowe do tworzenia trenowania sieci. ~~nałożenie zmierzonych parametrów na czyste nagranie studyjne.~~

System taki znajdzie zastosowanie między innymi jako wtyczka/rozszerzenie do programów obróbki dźwięku i będzie narzędziem efektywnym wykorzystywanym do realizacji utworów muzycznych. Narzędzie to byłoby w stanie zarówno nakładać efekty pogłosowe dopasowane do pożądanego pomieszczenia, jak i również usuwać niechciany wpływ pomieszczenia na nagranie – usuwanie pogłosu z nagrania.

Nagranie poddawane analizie w początkowej fazie projektowania algorytmu jest odpowiedzią impulsową pomieszczenia, co dokładniej opisane zostanie w dalszej części pracy.

Analiza próbek nagrań dźwiękowych w następnych rozdziałach odbywa się w kontekście ludzkiego postrzegania zachodzących zjawisk, aniżeli ich opisu w zakresie fizyki. Dobór parametrów charakteryzujących badany sygnał ma więc na celu odzwierciedlenie elementów, które dobrze reprezentują postrzeganie zmiany zawartości sygnału po jego cyfrowej obróbce, a także zmiany sygnału dźwiękowego w zależności od cech pomieszczenia.

Eksperymentalnym elementem, o który można by rozszerzyć działanie algorytmu jest uczenie maszynowe, które nadal jest często poruszonym tematem i ze względu na badawczy charakter pracy oraz podejście różniące się od zgodnych ze sztuką i normami pomiarami mogłoby stanowić interesujący wpływ na wynik działania programu. Wytrenowanie modeli sieci neuronowych na podstawie subiektywnych ocen ankietowanych osób mogłoby usprawnić działanie algorytmu w kwestii odzwierciedlenia zmian subiektywnych cech dźwięku.

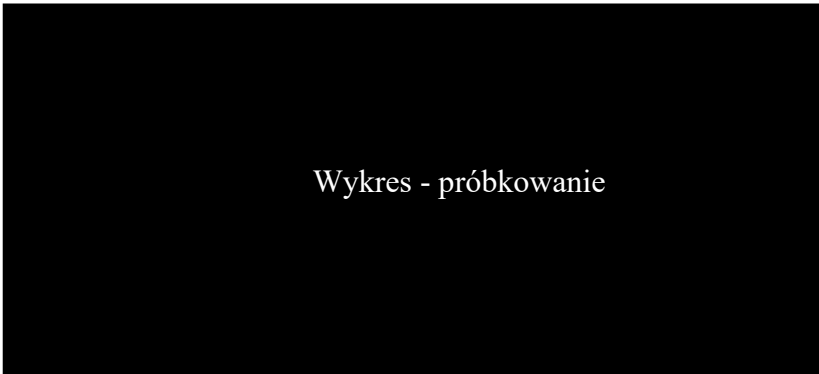
~~Parametryzacja nagrań w dalszym etapie pracy dyplomowej zostanie rozszerzona o uczenie maszynowe będące elementem eksperymantalnym oraz mające na celu usprawnienie obliczeń. Wpływ uczenia maszynowego na rozpoznanie charakteru nagrania (barwy itp.) zostanie zadany poprzez poddanie go ocenie subiektywnej przez słuchaczy.~~

3. Teoria

Cyfrowe przetwarzanie sygnałów jest ~~potężnym~~ narzędziem, umożliwiającym dokonywanie złożonych analiz sygnałów, które w technice analogowej stanowiłyby trudność, albo byłyby wręcz niemożliwe do realizacji. W połączeniu ze współczesnymi wysokopoziomowymi językami programowania stanowią środowisko, w którym jedynym ograniczeniem jest kreatywność programisty.

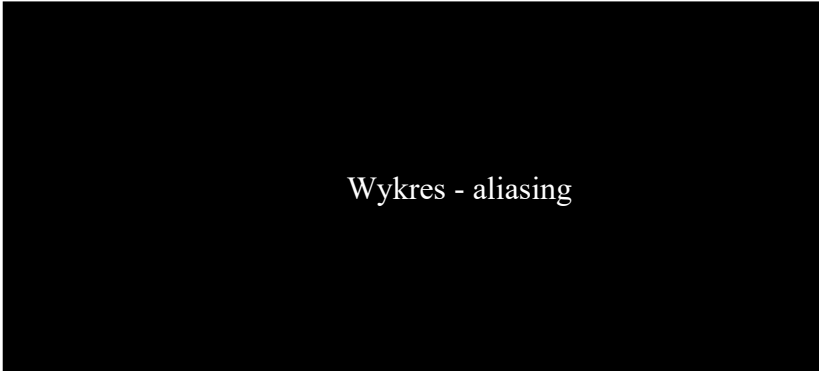
3.1. Podstawowe zagadnienia poruszane w pracy

Próbkowanie – dyskretyzacja sygnału w dziedzinach czasu i częstotliwości. Jest to niezbędny zabieg w technice cyfrowej, aby uzyskać skończoną liczbę danych umożliwiającą ich przetworzenie. Niewłaściwie dobrana częstotliwość próbkowania powoduje zjawisko aliasingu i w skrajnych przypadkach doprowadza do sytuacji, w której spróbkowany sygnał nie nadaje się do dalszego przetwarzania, gdyż nie odzwierciedla oryginału w taki sposób, w jaki oczekiwaliśmy.



Wykres - próbkowanie

Aliasing – aby sygnał cyfrowy został poprawnie odtworzony z postaci cyfrowej, najwyższa częstotliwość składowej sygnału musi równać się połowie częstotliwości próbkowania tegoż sygnału (częstotliwość Nyquista). W przeciwnym wypadku występuje zjawisko aliasingu, czyli błędnego odtworzenia spróbkowanego sygnału.



Wykres - aliasing

FFT – Fast Fourier Transform – Jest to algorytm służący do rozkładu sygnału na pojedyncze składowe częstotliwościowe. Jest to operacja matematyczna pozwalająca na przejście z dziedziny czasu na dziedzinę częstotliwości.

Dokonując operacji szybkiej transformaty Fouriera ważnym czynnikiem wpływającym na dokładność wyznaczonego widma sygnału jest ilość próbek w oknie czasowym FFT.

Kompresja dźwięku – dzieli się na dwie kategorie - kompresja stratna i bezstratna. Polega na zmniejszeniu zawartości redundantnych danych, bądź też usunięciu zbędnych danych w przypadku kompresji stratnej.

Format WAV - umożliwia zapis strumienia nieprzetworzonych danych. Zawiera on sygnały o wyższych częstotliwościach niż sygnał zakodowany w formacie MP3, a także zawiera dźwięki które są niesłyszane przez ludzkie ucho, takie jak tony zamaskowane tonem o wyższej amplitudzie, które kodowanie MP3 usuwa.

Czas pogłosu - definiuje się jako zanik energii o 60dB. Jest to tysiąckrotny zanik poziomu ciśnienia akustycznego, co zapisuje się:

$$10^{\frac{60 \text{ dB}}{20}} = 1000.$$

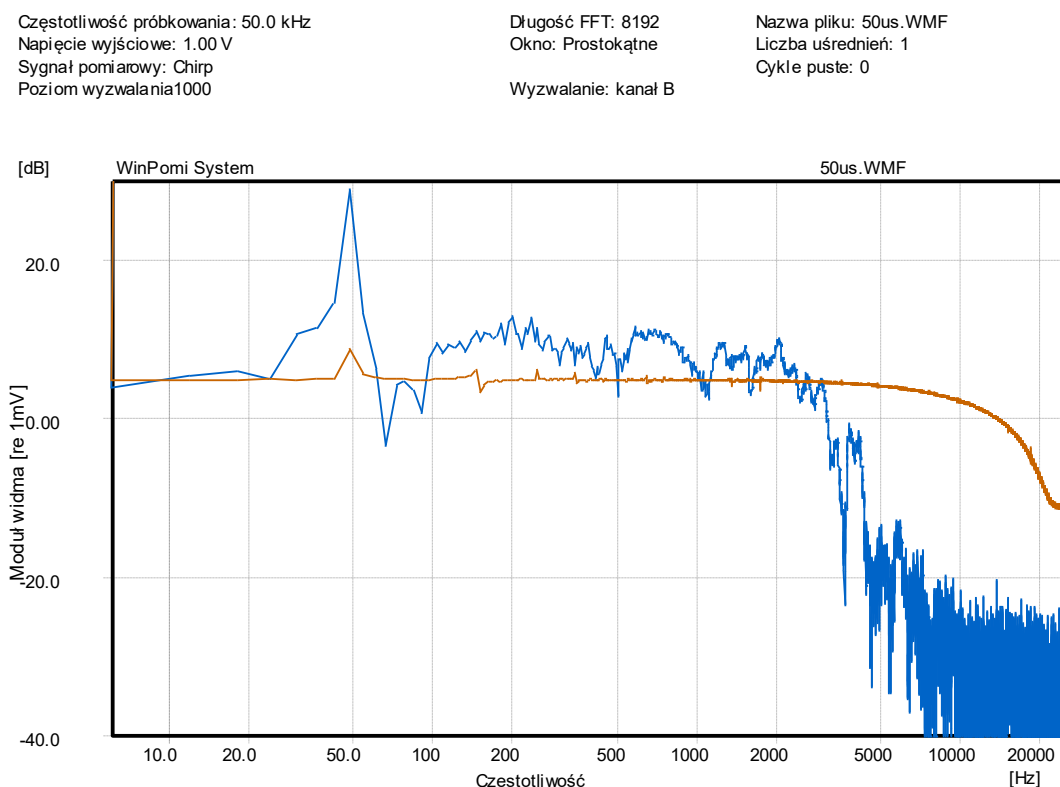
Parametr T30 - jest to czas, po jakim poziom energii spadnie o 60dB względem stanu początkowego, wyznaczany na podstawie nachylenia krzywej zaniku poziomu energii w zakresie 30dB, co umożliwia wyznaczenie czasu pogłosu nawet w przypadku, gdy mamy do czynienia z szumami.

.... więcej

3.2. Charakterystyka częstotliwościowa w zależności od długości impulsu.

Do pomiaru czasu pogłosu metodą impulsową należy uwzględnić rodzaj źródła tegoż sygnału. W poniższych przykładach zaprezentowano sygnał impulsowy z generatora odtwarzany na głośnikach. Innym, podobnym źródłem sygnału impulsowego mogą także być: pękający balon czy wystrzał z pistoletu.

W zależności od charakteru źródła, możemy mieć do czynienia z bardzo wąskim pasmem użytecznych częstotliwości, bądź też niskim poziomem sygnału, gdy sygnał ma bardzo krótki czas trwania (co zaprezentowano na poniższych przykładach):

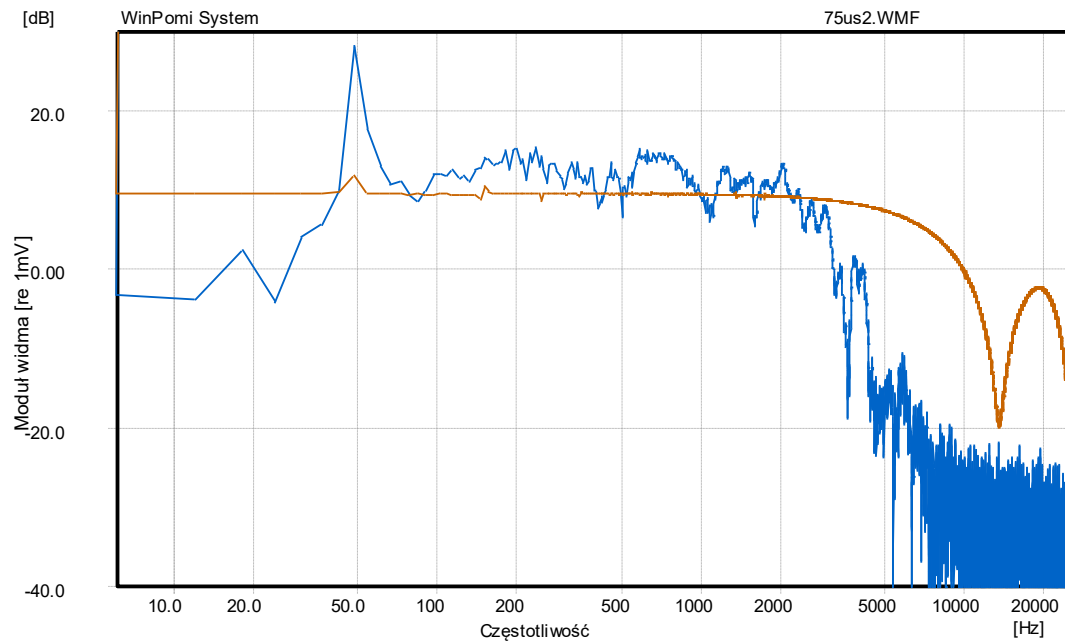


• Rysunek 1 Charakterystyka częstotliwościowa dla impulsu 50 μ s

Częstotliwość próbkowania: 50.0 kHz
Napięcie wyjściowe: 1.00 V
Sygnał pomiarowy: Chirp
Poziom wyzwalania: 1000

Długość FFT: 8192
Okno: Prostokątne
Wyzwalanie: kanał B

Nazwa pliku: 75us2.WMF
Liczba uśrednień: 1
Cykle puste: 0

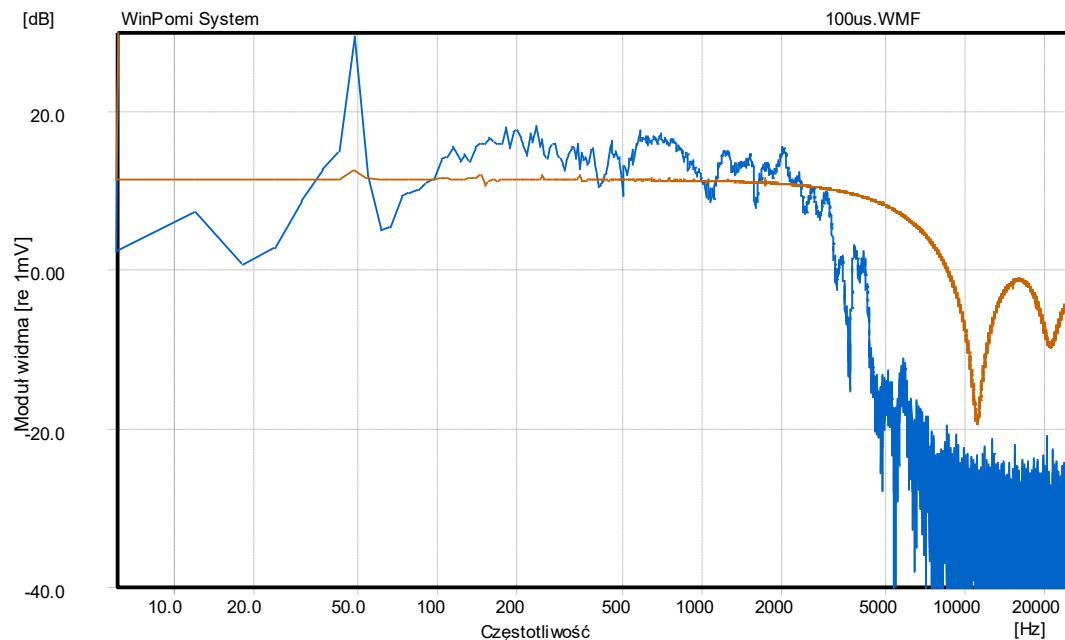


Rysunek 2 Charakterystyka częstotliwościowa dla impulsu 75 μ s

Częstotliwość próbkowania: 50.0 kHz
Napięcie wyjściowe: 1.00 V
Sygnał pomiarowy: Chirp
Poziom wyzwalania: 1000

Długość FFT: 8192
Okno: Prostokątne
Wyzwalanie: kanał B

Nazwa pliku: 100us.WMF
Liczba uśrednień: 1
Cykle puste: 0

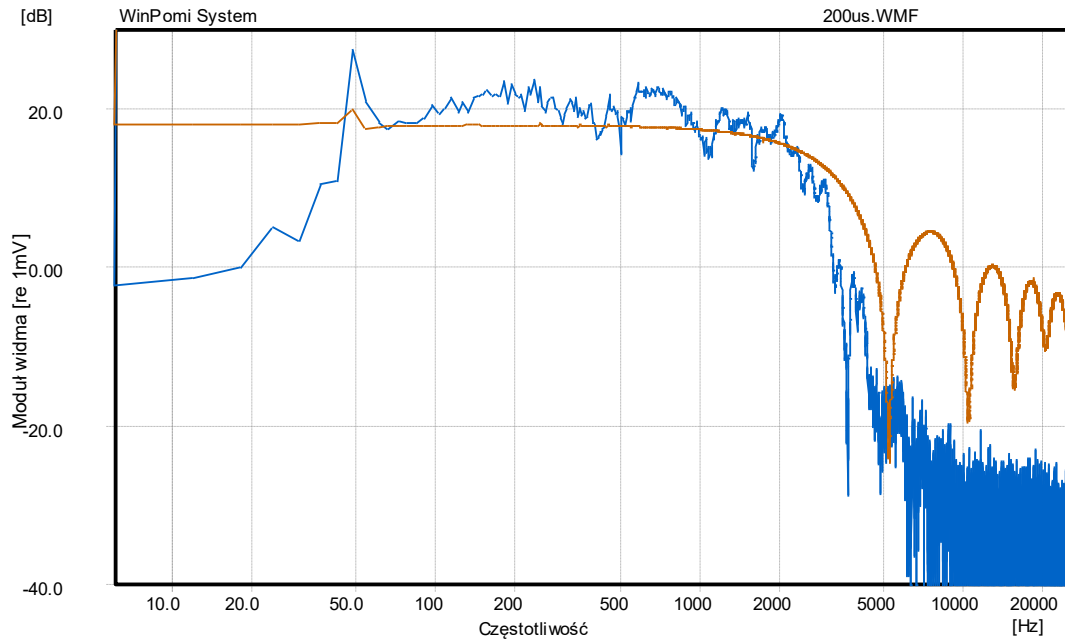


Rysunek 3 Charakterystyka częstotliwościowa dla impulsu 100 μ s

Częstotliwość próbkowania: 50.0 kHz
Napięcie wyjściowe: 1.00 V
Sygnał pomiarowy: Chirp
Poziom wyzwalania: 1000

Długość FFT: 8192
Okno: Prostokątne
Wyzwalanie: kanał B

Nazwa pliku: 200us.WMF
Liczba uśrednień: 1
Cykle puste: 0

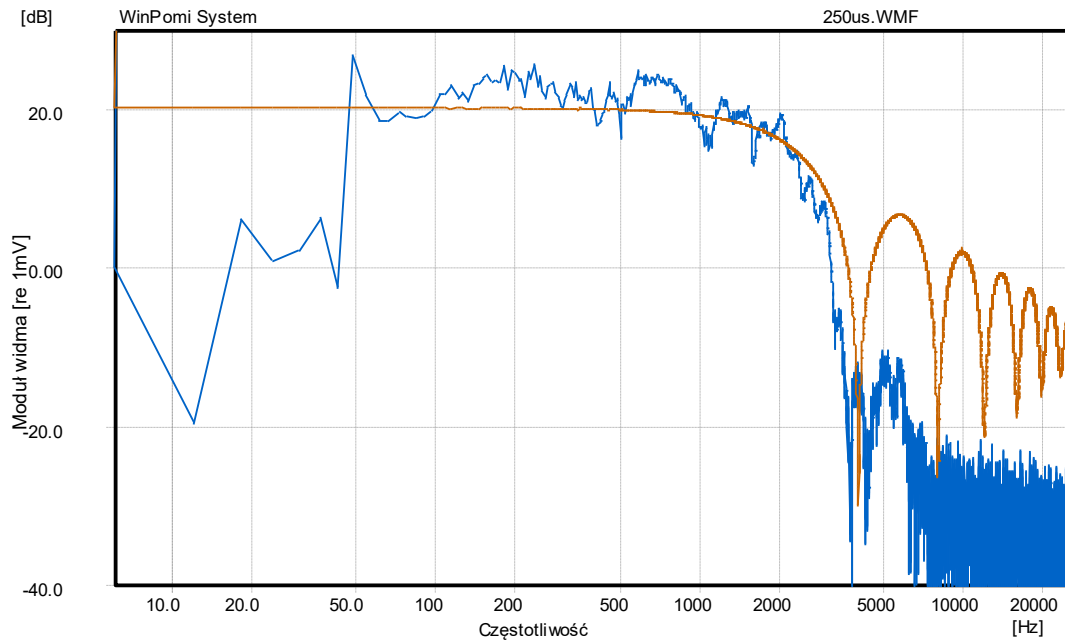


Rysunek 4 Charakterystyka częstotliwościowa dla impulsu 200 μ s

Częstotliwość próbkowania: 50.0 kHz
Napięcie wyjściowe: 1.00 V
Sygnał pomiarowy: Chirp
Poziom wyzwalania: 1000

Długość FFT: 8192
Okno: Prostokątne
Wyzwalanie: kanał B

Nazwa pliku: 250us.WMF
Liczba uśrednień: 1
Cykle puste: 0

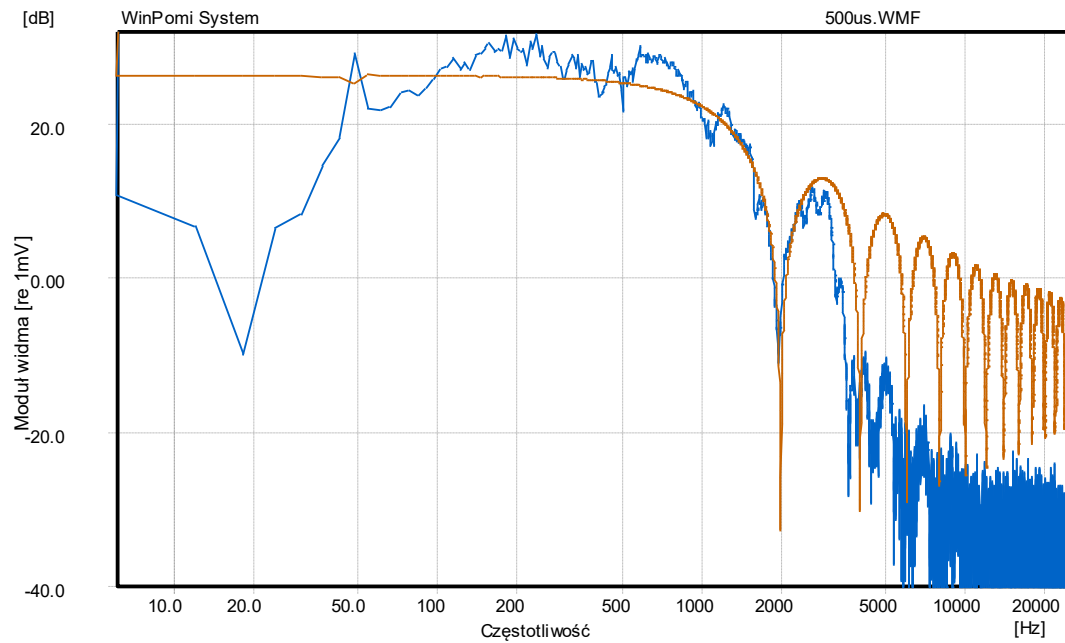


Rysunek 5 Charakterystyka częstotliwościowa dla impulsu 250 μ s

Częstotliwość próbkowania: 50.0 kHz
Napięcie wyjściowe: 1.00 V
Sygnał pomiarowy: Chirp
Poziom wyzwalania: 1000

Długość FFT: 8192
Okno: Prostokątne
Wyzwalanie: kanał B

Nazwa pliku: 500us.WMF
Liczba uśrednień: 1
Cykle puste: 0

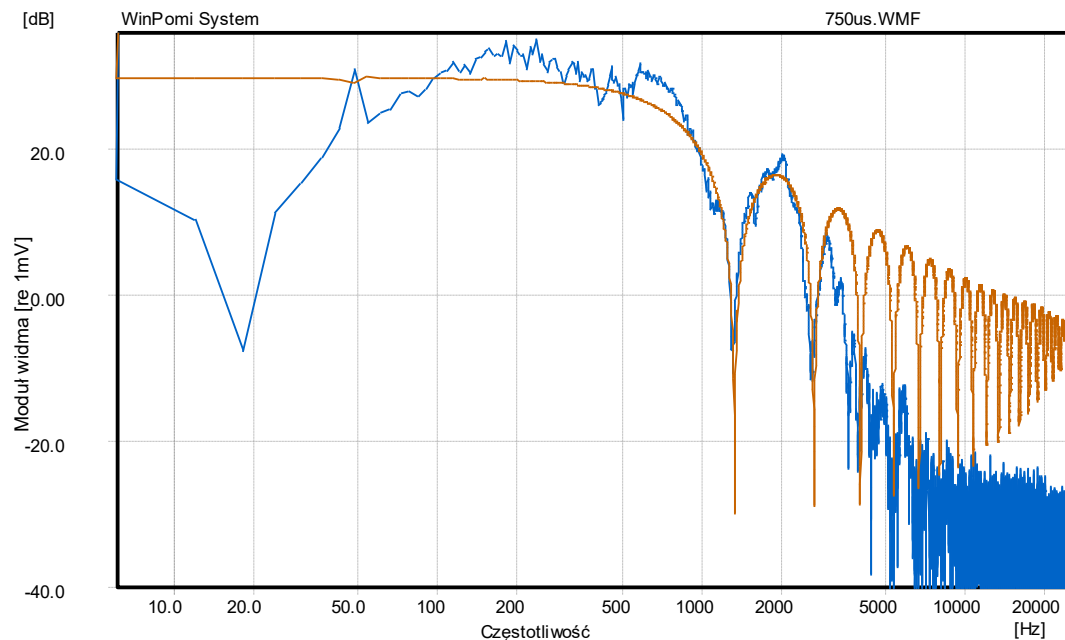


Rysunek 6 Charakterystyka częstotliwościowa dla impulsu 500 μ s

Częstotliwość próbkowania: 50.0 kHz
Napięcie wyjściowe: 1.00 V
Sygnał pomiarowy: Chirp
Poziom wyzwalania: 1000

Długość FFT: 8192
Okno: Prostokątne
Wyzwalanie: kanał B

Nazwa pliku: 750us.WMF
Liczba uśrednień: 1
Cykle puste: 0

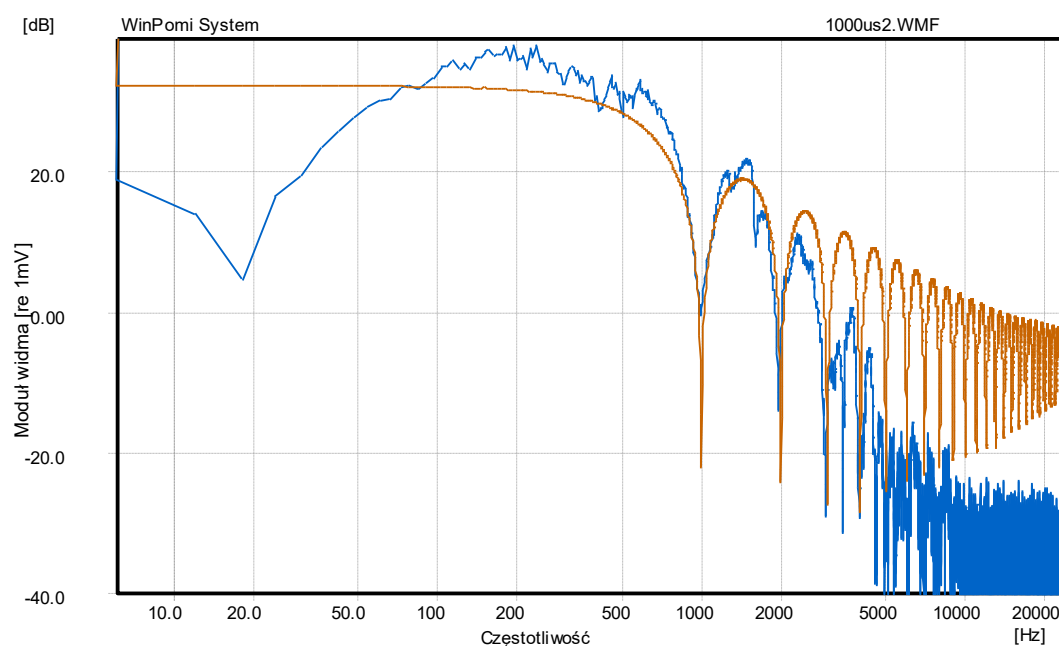


Rysunek 7 Charakterystyka częstotliwościowa dla impulsu 750 μ s

Częstotliwość próbkowania: 50.0 kHz
Napięcie wyjściowe: 1.00 V
Sygnał pomiarowy: Chirp
Poziom wyzwalania: 1000

Długość FFT: 8192
Okno: Prostokątne
Wyzwalanie: kanał B

Nazwa pliku: 1000us2.WMF
Liczba uśrednień: 1
Cykle puste: 0



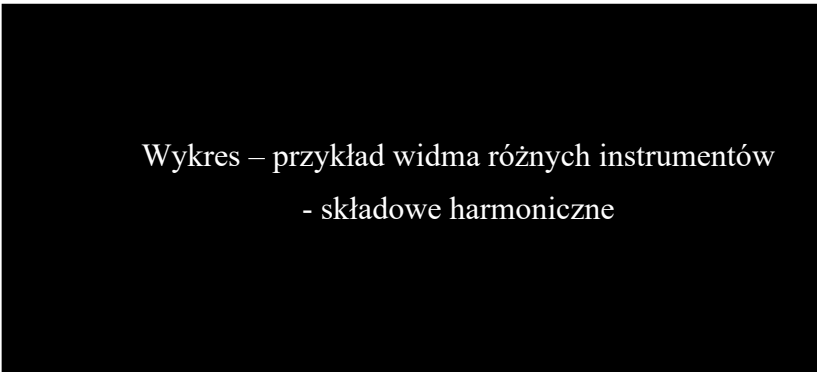
Rysunek 8 Charakterystyka częstotliwościowa dla impulsu $1000 \mu s$

Omówienie do każdego/kilku z wykresów

3.3. Barwa dźwięku

Barwa dźwięku jest subiektywną cechą, którą każda osoba może postrzegać w znacznie różniący się od siebie sposób, stąd też ciężko zdefiniować jest dokładną wartość tego parametru.

Instrumenty muzyczne potrafią wytworzyć ton o tej samej wysokości dźwięku, niemniej jednak dźwięki te będzie rozróżniać właśnie barwa, na którą wpływ ma między innymi szerokość widma emitowanego sygnału, a także zawartość i amplituda składowych harmoniczných w tymże sygnale.



Wykres – przykład widma różnych instrumentów
- składowe harmoniczne

3.3.a. Spectral centroid

Spectral centroid, czyli środek ciężkości charakterystyki częstotliwościowej sygnału jest parametrem, który został użyty w algorytmie w celu aproksymacji barwy dźwięku rozumianej jako częstotliwości dominujące w danym nagraniu w określonym odcinku czasu.

4. Projekt aplikacji

Do implementacji projektu wybrano język Python w wersji 3,7, który charakteryzuje się szerokim zakresem zastosowań ze względu na wysokopoziomą składnię, pozwalającą na szybką implementację złożonych czy rozbudowanych aplikacji. Wsparcie w postaci różnorodnych i specjalistycznych bibliotek dla tego języka umożliwia znacząco szybsze rozwiązywanie zaawansowanych problemów, które na przykład w przypadku języka C++ wymagałyby wielokrotnie więcej nakładu pracy (co mogłoby być niewspółmierne do zalet płynących z zastosowania C++). Wersja 3,7 zapewnia wsparcie w możliwym późniejszym rozwoju aplikacji.

4.1. Możliwości

Zrozumiałość mowy w danym pomieszczeniu najbardziej zależna jest od powstających wczesnych odbić, które definiują stosunek oryginalnego sygnału do jego ogona pogłosowego. Analizując odpowiedź impulsową pomieszczenia możemy więc analizować sygnał pod tym kątem.



Wykres – energia w pierwszych 50ms vs ogon


Dokładniejszą metodą analizy powstających odbić jest zastosowanie sygnału sinusoidalnego w postaci krótkiego impulsu, który można następnie przeanalizować dokonując korelacji z czystą reprezentacją tego sygnału.



Wykres – korelacja IR <-> sin

PODDZIAŁ Usuwanie pogłosu z nagrania przy pomocy pomiaru odpowiedzi impulsowej pomieszczenia.

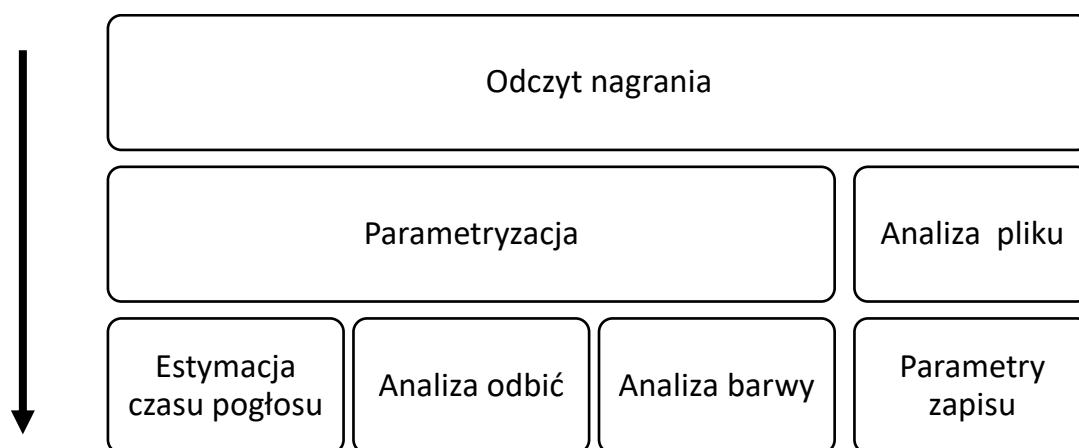
[Jeszcze niezaimplementowane]



Ilustracja koncepcji procesu

Omówienie

4.2. Schemat działania



Wstępne omówienie – więcej w dalszej części

4.3. Implementacja

W pierwszym kroku algorytm dokonuje odczytu pliku w formacie *wave*, który umożliwia zapis nagrania bez kompresji. ~~We wstępnej analizie badane są:~~

~~ilość kanałów,~~

~~częstotliwość próbkowania,~~

~~ilość próbek.~~

Do wykonania tego etapu utworzono klasę *WaveFile*, która inicjowana jest adresem do pliku:

```
class WaveFile:
    PATH = str()

    def __init__(self, path):
        self.PATH = path
        self.FILE = wave.open(self.PATH)
        [self.CHANNELS, self.SAMP_WIDTH, self.FRAMERATE, self.NFRAMES, *self.rest]
= self.FILE.getparams()
        self.RAW = np.fromstring(self.FILE.readframes(-1), dtype='int32')
        self.RAW = self.RAW/max(self.RAW)
```

Przykład utworzenia instancji klasy:

```
orig_file = WaveFile('./PathToFile/orig.wav')
```

Opis wykorzystanych zmiennych:

- PATH - ścieżka do pliku,
- FILE - obiekt modułu *wave*,
- CHANNELS - liczba kanałów,
- SAMP_WIDTH - ilość bajtów dla jednej próbki (przeważnie jest to 2),
- FRAMERATE - częstotliwość próbkowania,
- NFRAMES - liczba próbek,
- RAW - nieprzetworzony zapis pliku *wave*.

Odczyt wartości próbek zapisany do zmiennej *RAW* (będącej jednowymiarową tabelą typu *int32*¹), która w następnym kroku algorytmu zostaje znormalizowana (w wyniku operacji dzielenia typ zmiennej ulega rzutowaniu na typ *float*).

Celem wyżej wymienionych analiz pliku *wave* jest przygotowanie odpowiednich kroków w dalszej części algorytmu.

¹ 32 bitowa liczba całkowita z zakresu od -2^{31} do $2^{31} - 1$.

4.3.a. Pogłos - Badanie odpowiedzi impulsowej

Najpierw obliczany jest kwadrat dla każdej wartości próbki:

```
for i, val in enumerate(frames):
    f_squared[i] = do_square(val)
```

Następnie odwrócono kolejność próbek:

```
f_squared = np.flip(f_squared)
```

Po czym dokonano całkowania i utworzono oś czasu X:

```
integral = integrate(f_squared)
time = np.linspace(0, len(integral)/FP, num=len(integral))
```

Najpierw refactor kodu, a potem OPISAĆ:

```
db30o = x[y.searchsorted(-30, 'left')]
db30 = round(db30o*1000*CHOP)
db0o = x[y.searchsorted(-0.2, 'right')]
db0 = round(db0o*1000*CHOP)
print("0dB--> at: " + str(db0) + "ms")
print("-30dB-> at: " + str(db30) + "ms")
print("-60dB-> at: " + str(db30+(db30-db0)) + "ms [extrapolated]")
print("T30----->: " + str((db30-db0)*2) + "ms")

x1 = int((db30o)*FP) # spadek o 30dB
x2 = int((db0o)*FP) # odniesienie
x3 = int((db0o+((db30o-db0o)*2))*FP) # spadek o 60dB szacowany
```

Na pliku reprezentującym odpowiedź impulsową pomieszczenia przeprowadzana zostaje operacja całkowania.

$$E(t) = \int_{t+T_0}^t p^2(\tau) d(-\tau)$$

Równanie 1 PN-EN ISO 3382

Z wykresu krzywej zaniku poziomu energii szacowany jest spadek o 60dB na podstawie zakresu 30dB [parametr T30].

4.3.a. Przykładowy pomiar zaniku poziomu energii w pomieszczeniu.

Poniżej zaprezentowano wzorcowy pomiar czasu pogłosu metodą impulsową wykonany na profesjonalnym sprzęcie, zgodnie z normą *PN-EN ISO 3382*.

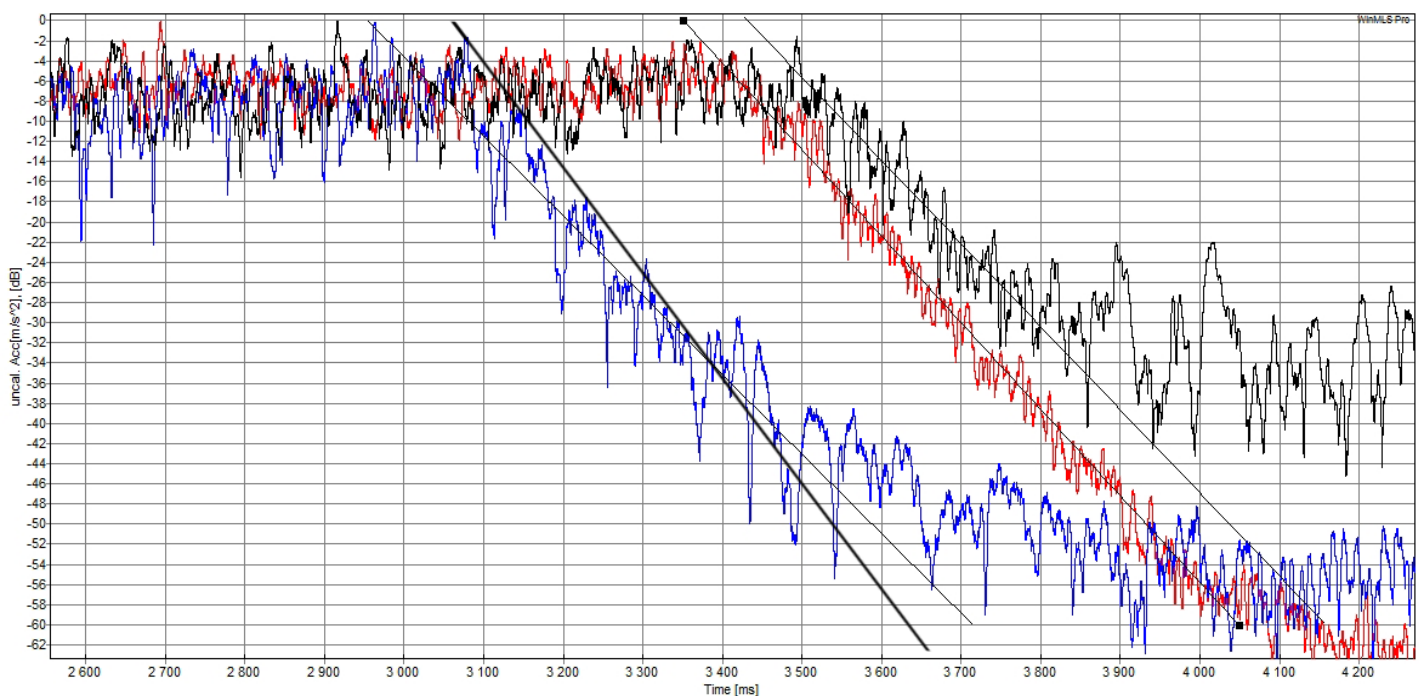
Nagrania wykonano w trzech miejscach położenia mikrofonu oraz trzech miejscach położenia źródła dźwięku (co daje razem 9 nagrań na jedno pomieszczenie).

Tabela 1 Pomiar wymiarów pomieszczenia.

wymiar pomieszczenia [m]	
długość	5,92
szerokość	5,40
wysokość	3,40

Tabela 2 Współczynniki pochłaniania wybranych elementów pomieszczenia.

	wsp. pochłaniania	powierzchnia [m ²]
drzwi	0,18	2,03
meble	0,17	7,00
okno	0,18	7,16
ściany	0,02	59,04
drzwi	0,17	1,76
podłoga	0,31	31,97
sufit	0,05	31,97

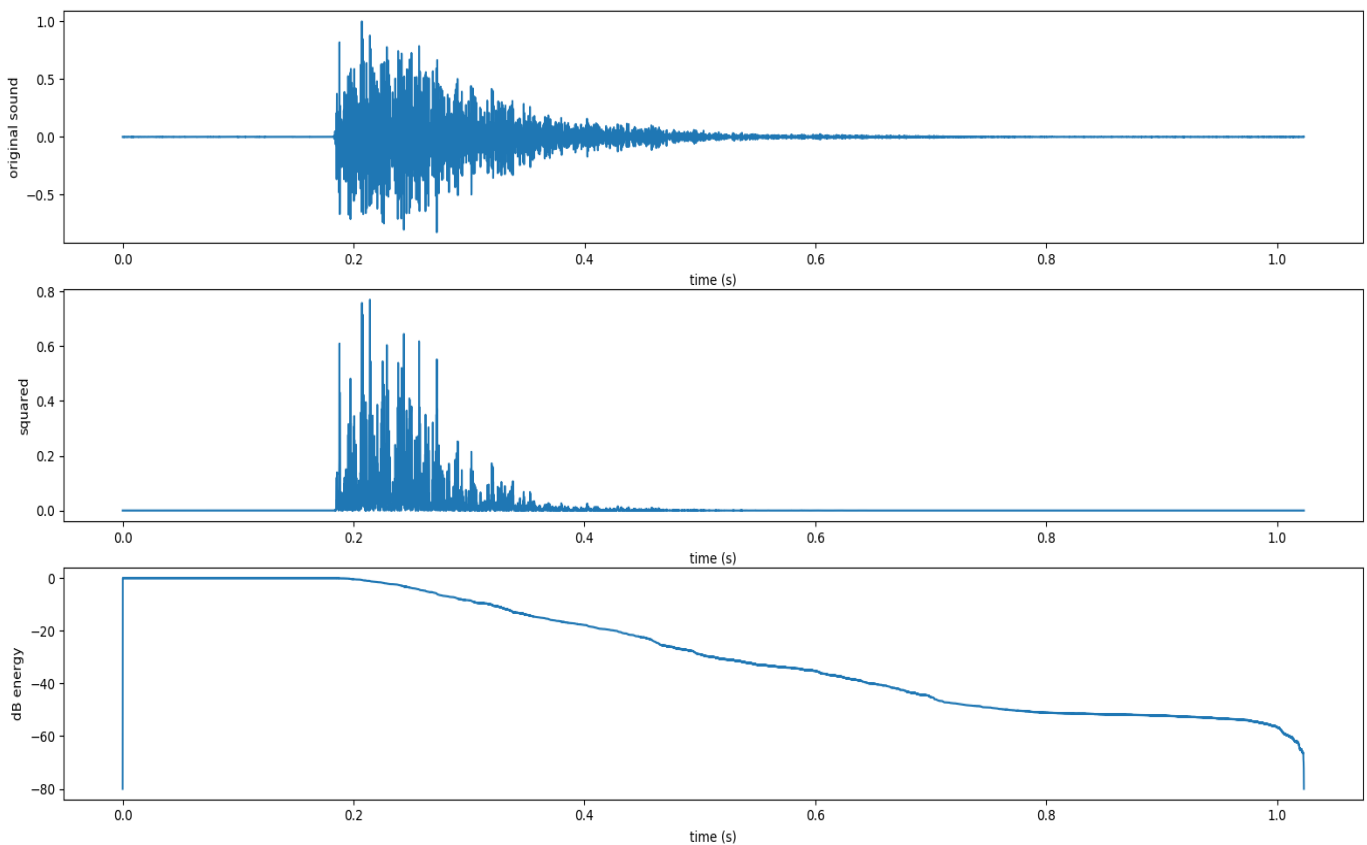


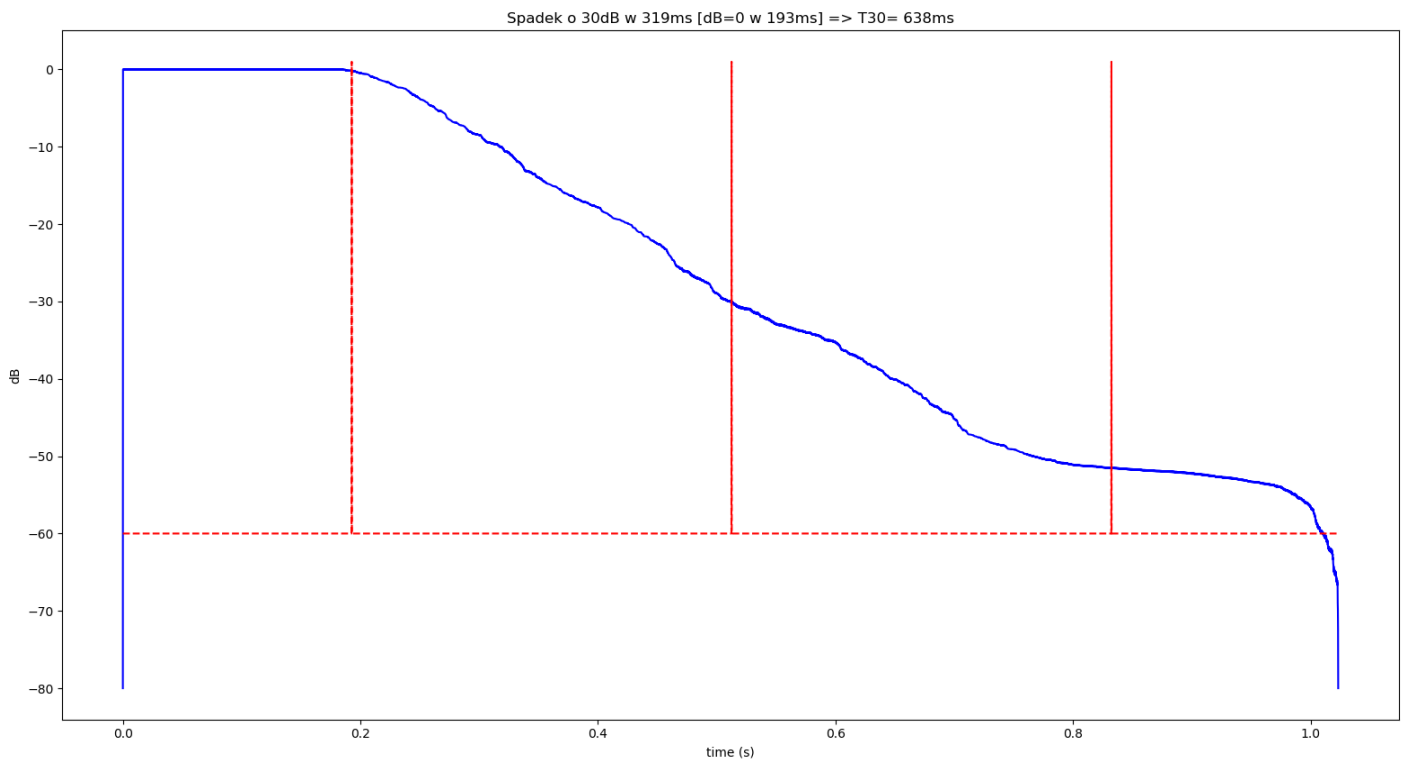
Wykres 1. Krzywe zaniku poziomu energii [czerwona - 2kHz, niebieska - 500Hz, czarna – bez filtra]

4.3.b. Pomiar pogłosu poprzez omawiany program

Algorytm wyliczający czas pogłosu bazuje na metodzie impulsowej. Do wykonania nagrań odpowiedzi impulsowej jako użyto różnego źródła dźwięku różnego rodzaju, między innymi – pękający balon, trzaśnięcie, kłaśnięcie, aby oszacować wpływ jakości nagrania odpowiedzi impulsowej na późniejszą operację splotu [całe badanie ma charakter subiektywnego postrzegania działania algorytmu, aniżeli odwzorowywania pomiarów zgodnie ze sztuką].

Do zminimalizowania poziomu tła w nagraniach dokonano analizy w pasmach częstotliwościowych [jakich? – pasma +spadki] przy użyciu filtrów cyfrowych, co pozwoliło na uzyskanie większego zakresu (około 30-50dB) zaniku poziomu dźwięku pozwalającego oszacować jego spadek o 60dB.





Algorytm określił kolejne wartości:

Poziom [dB]	Czas[ms]
0	193,0
-30	512,0
-60 [T30]	638,0

Gdzie poziom w decybelach jest mierzony względem najwyższej odnotowanej amplitudy sygnału

$$L = 10 \log_{10} \left(\frac{I}{I_0} \right), I_0 = V_{max}$$

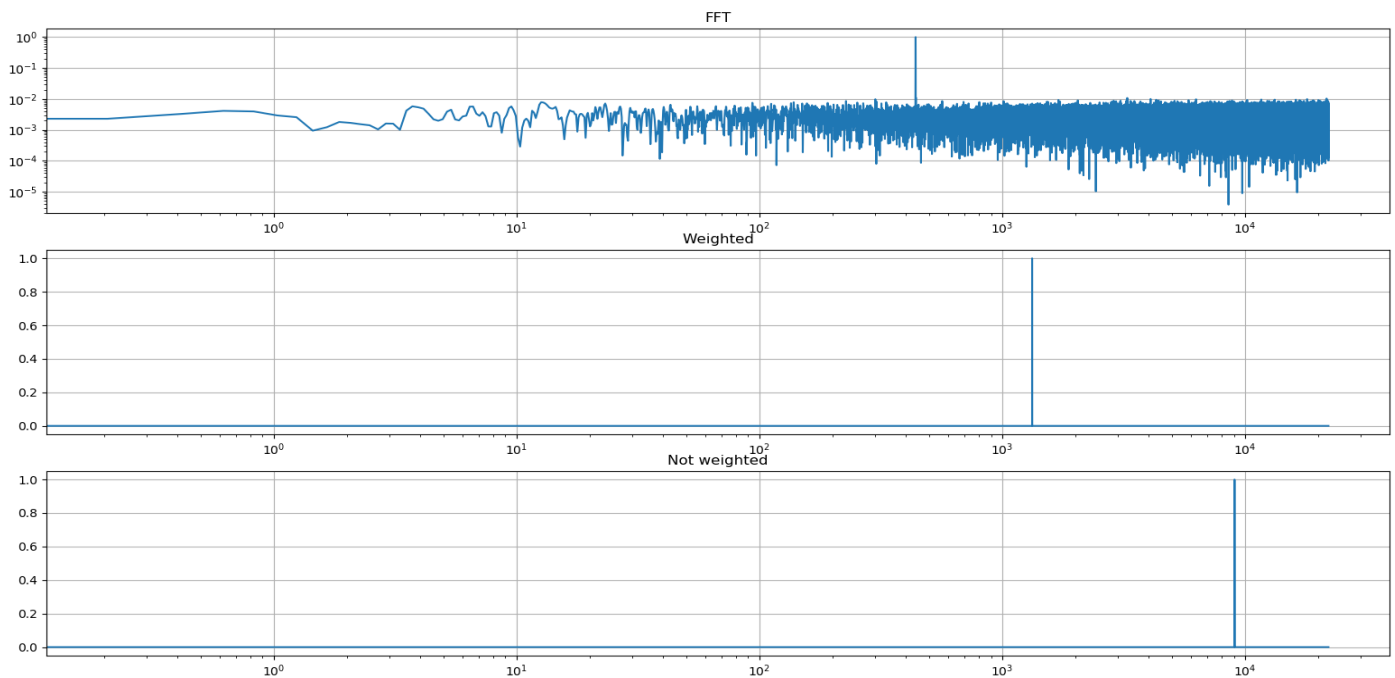
Z założeń projektowych algorytmu, przy analizie nagrania pominięto częstotliwości mniejsze, jak 120Hz ze względu na dodatkowe trudności występujące w analizie tychże częstotliwości, takie jak rezonanse osiowe, styczne i skośne, a także wpływ jakości mikrofonu na pomiar częstotliwości w tym zakresie. Zjawiska te mogą wpływać na dodatkowe błędy, które utrudnią analizę działania projektowanego algorytmu, z tego powodu zostaną one pominięte, a analiza zostanie przeprowadzana w przypadkach „idealnych”, czyli niestwarzających dodatkowych problemów.

4.3.c. Analiza zmiany barwy

Do algorytmu wyznaczającego parametr centroidy częstotliwościowej dodano ważenie amplitudy zgodnie z założeniami prawa Webera Fechnera, ze względu na to, iż rozdzielczość *FFT* na wykresie wynosi 10Hz , co każdy punkt. Stąd też występuje zagęszczenie pomiarów dla wyższych częstotliwości dla skali logarytmicznej (która reprezentuje sposób postrzegana wrażeń dźwiękowych przez człowieka). Stąd, aby uzyskać bardziej naturalny dla ludzkiego ucha wynik działania tego algorytmu dodano ważenie wartości amplitudy *FFT* w odniesieniu do częstotliwości.

Przykłady

Przykład na wykresie [sinus 440Hz + szum biały] (kolejno *FFT* sygnału, wartość *SC* ważone, wartość *SC* nieważone):

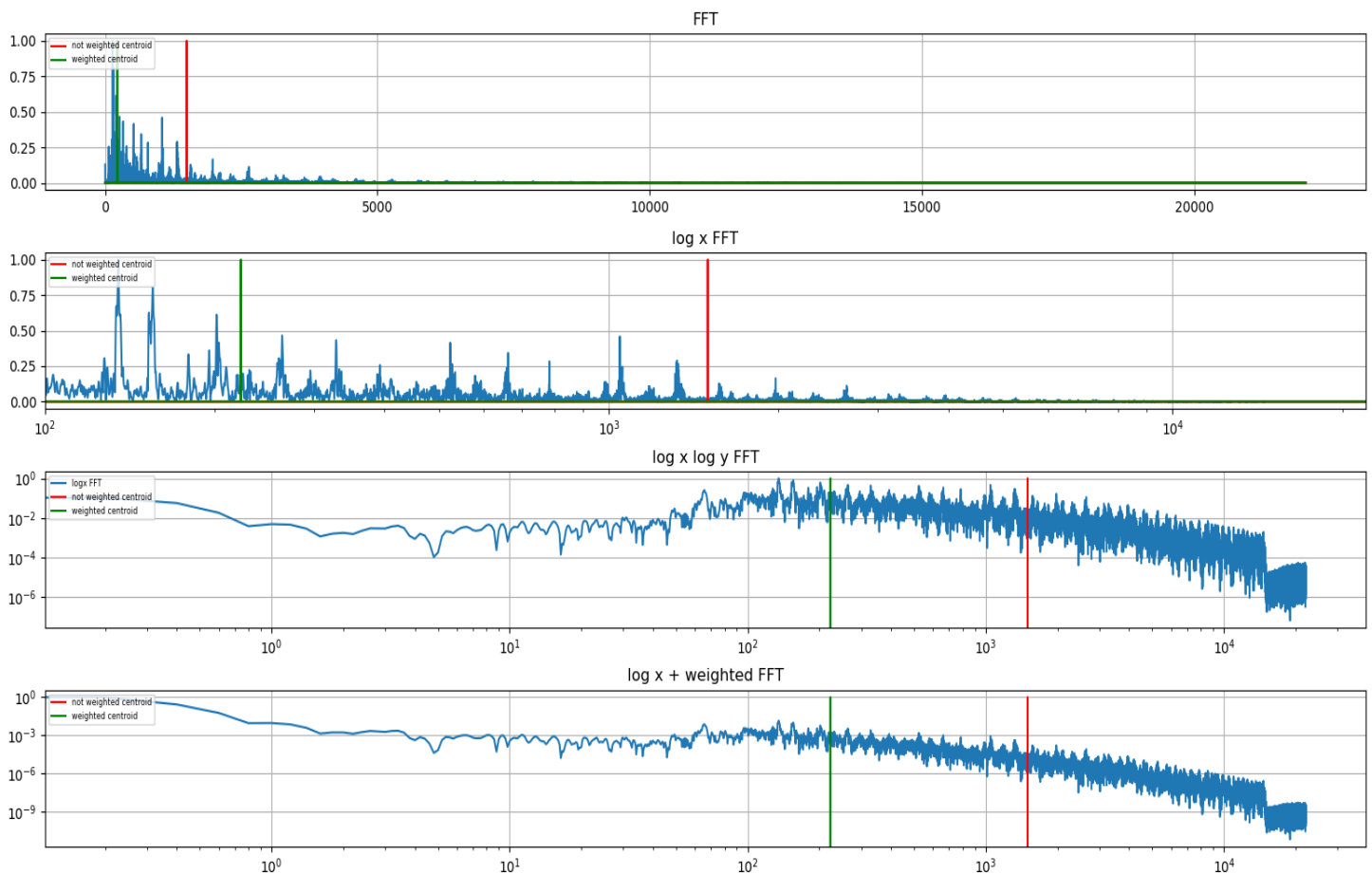


Kolejny przykład – utwór muzyczny

Kolejno:

- FFT w skali liniowej (X lin Y lin) [wartości Y nieważone]
- FFT w skali logarytmicznej (X log Y lin) [wartości Y nieważone]
- FFT X log i Y log [wartości Y nieważone]
- FFT X log i Y log [wartości Y ważne]

Legenda:



Poprawka ta widocznie udoskonala znajdowanie środka wagi widma.

4.3.d. Transformacja sygnału

Nałożenie zmierzonej odpowiedzi impulsowej na sygnał nagrany w warunkach zbliżonych do warunków studyjnych (niezauważalny pogłos oraz brak zniekształceń sygnału).

W celu przyspieszenia obliczeń splot realizowany jest jako mnożenie w dziedzinie częstotliwości.

$$y(t) = h(t) * x(t) \equiv Y(s) = H(s) \cdot X(s)$$

```
def nextpow2(L):  
    N = 2  
    while N < L: N *= 2  
    return N  
  
def convolution(x, h):  
  
    L = len(h) + len(x) - 1 # linear convolution length  
    N = nextpow2(L)  
  
    H = np.fft.rfft(h, N) # Fourier transform of the impulse  
    X = np.fft.rfft(x, N) # Fourier transform of the input signal  
  
    Y = H * X # spectral multiplication  
    y = np.fft.irfft(Y) # time domain again  
  
    y = np.array(y/(max(y)*1.001), dtype='float32')  
    return y
```

```
splot = convolution(orig_file.RAW, IR_file.RAW)  
plot(orig_file.RAW, IR_file.RAW, splot)
```

Do łatwej analizy wyniku działania programu użyto biblioteki *matplotlib* dostarczającej łatwych narzędzi do tworzenia wykresów. Zaimplementowano funkcje *plot* oraz *plot_signals* umożliwiające tworzenie wykresów za pomocą wcześniej wspomnianej biblioteki bezpośrednio na instancjach klasy *Signal*.

```
def plot(toPlot, *args):  
  
    print(len(args))  
    N_PLOTS = 1 + len(args)  
  
    plt.subplot(N_PLOTS, 1, 1)  
    plt.plot(timeLine(toPlot, 44100), toPlot)  
  
    for index, pl in enumerate(args):  
  
        plt.subplot(N_PLOTS, 1, index+2)  
        plt.plot(pl)  
  
    plt.show()
```

```
def plot_signals(*args: Signal):  
  
    for i, arg in enumerate(args):  
        plt.subplot(len(args), 1, i+1)  
        plt.title(arg.title)  
        if arg.timedomain:  
            plt.xlabel('time [s]')  
            plt.plot(arg.time, arg.data)  
        else:  
            plt.plot(arg.data)  
            plt.xlabel('frequency [Hz]')  
    plt.show()
```

```
orig_file = WaveFile(ORIG)  
IR_file = WaveFile(IR)  
  
sig_1 = Signal(file=orig_file,  
               title="Original file",  
               timeDomain=True)  
sig_2 = Signal(file=IR_file,  
               title="Impulse response",  
               timeDomain=True)  
  
plot_signals(sig_1, sig_2)
```


4.3.e. Zapis nowego pliku

```
wav_file = wave.open("file2.wav", 'w')
```

```
nchannels = 1  
sampwidth = 2  
framerate = 44100  
nframes = len(splot)  
comptype = "NONE"  
compname = "not compressed"
```

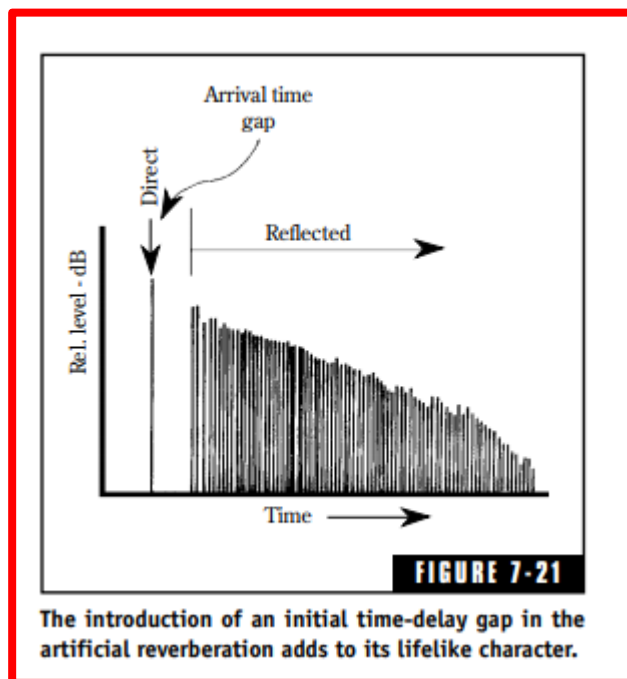
```
wav_file.setparams((nchannels,  
                    sampwidth,  
                    framerate,  
                    nframes,  
                    comptype,  
                    compname  
                    ))
```

```
wav_file.writeframes(np.array(splot, dtype='float32'))  
wav_file.close()
```

4.4. Pogłos

Oddanie naturalności brzmienia, w kontekście zjawiska pogłosu wymaga między innymi rozpatrzenia jednego z ważnych elementów, jakim jest opóźnienie pomiędzy źródłem dźwięku, a pierwszym odbiciem, które to nadaje wrażenie wielkości pomieszczenia.

Fala dźwiękowa pokonując bezpośrednią drogę dociera najszybciej do słuchacza, następnie docierają wczesne odbicia. Opóźnienie to mierzone jest przez algorytm [fragment kodu z omówieniem].

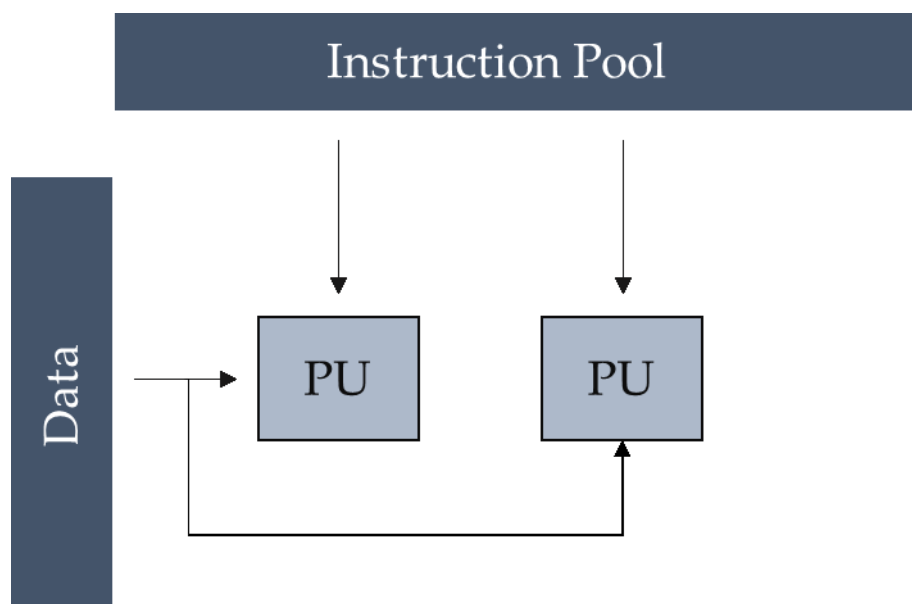


Kolejnym elementem jest gęstość ech występujących w ogonie pogłosowym, która jest wymagana, aby dźwięk wydawał się naturalny i aby nie wystąpił efekt flutter. Zgodnie z badaniami Schroedera szacuje się, że gęstość ta wynosi około 1000 ech na sekundę.

4.5. Równoległość parametryzowania

W przypadku analizy plików o większej objętości czas obliczeń można skrócić stosując zrównoleglenie procesu parametryzacji. Obecnie równoległość przetwarzania danych jest standardem dla profesjonalnych aplikacji komputerowych, gdyż są one w stanie wykorzystać potencjał, który zapewnia wielordzeniowość dzisiejszych komputerów

Proponującą wykorzystania równoległości jest architektura „Multiple Instructions, Single Data”.



Wybór tej metody uzasadniam poprzez charakter projektowanej aplikacji, która analizuje w danym czasie wyłącznie jeden plik, stąd też nie możemy zastosować architektury „*Single Instruction, Multiple Data*”, która jest częściej spotykanym podejściem.

Algorytmy analizy sygnału w czasie, jak i w dziedzinie częstotliwości dokonują operacji i przekształceń wszystkich próbek, stąd też w większości mają te same czasy potrzebne na ich wykonanie. Dzięki zrównolegleniu uzyskujemy zamiast $x[s] \cdot n$ czasu analizy, po prostu $x[s]$.

Przykład kodu zgodnego z powyższymi założeniami, realizującego równoległą parametryzację:

```
for (i, [param, func]) in enumerate(function_dict.items()): # execute each function from dict

    print(i,param, func)

    pool.apply_async(extract_attributes, args=(i, func, chunks, tot, param))

pool.close()
pool.join()
```

Dla słownika *function_dict*, wszystkie jego przedmioty (*ang. items*), czyli funkcje wykonywane są na sygnale, ale w osobnych wątkach wywoływanych za pomocą instrukcji *pool.apply_async()* powodujące asynchroniczne wykonanie funkcji parametryzującej.

```
function_dict = {
    "SR": spectral_rolloff,
    "SF": spectral_flatness,
    "SC": spectral_centroid,
    "MFCC": get_mfcc,
    "EIB": energy_in_bands,
    "Peaks": get_peak,
    "SNR": get_snr,
    "RMS": rootmean,
    "ZC": zero_cross
}
```

Powyżej zaprezentowano przykład słownika zawierającego spis funkcji, które należy wykonać na sygnale. Nazwa funkcji w języku Python jest wskaźnikiem na adres tej funkcji, co umożliwia wywoływanie w pętli różnej funkcji przy tej samej zmiennej.

```
def extract_attributes(pron_num, f, data, params, p_name): #, rdy):
    print('Created separated process for: ', p_name)
    start = time.time()
    temp = []
    for chunk in data: # for each chunk of data
        temp.append(f(chunk))
    params[pron_num]=temp
    print('process ended for: %s in %0.2fs' % (p_name, time.time()-start))
```

Ostatnim elementem jest sama funkcja bezpośrednio wywoływana w pętli i wykonująca daną instrukcję ze słownika funkcji parametryzującej. Pełni ona rolę „Wrappera” poprzez iteracyjne wykonywanie analizy i zapamiętywanie wyniku w tablicy i ostatecznie zapisywanie do pamięci współdzielonej reprezentowanej poprzez zmienną *params*.

5. Uczenie maszynowe

Będzie tylko wspomniane, jako projekt, ale bez implementacji, boadź też ze szkicem jak mogłaby wyglądać implementacja, bo mam pomysły.

5.1. Wstęp

W celach eksperymentalnych założono implementację algorytmów uczenia maszynowego do wcześniej omówionego programu.

Uczenie maszynowe pozwala na optymalizację oraz znajdowanie rozwiązań dla złożonych problemów w mniej konwencjonalny sposób, niż wierne sztuce metody opisywane w normach.

6. Podsumowanie

Np. zestawienie - wykres zmiany barwy wejście a wyjście

Ocena jakości działania programu

Coś o możliwym zastosowaniu uczenia maszynowego i możliwych korzyści

