Audio Projekt1

Aleksander Malinowski, Damian Skowroński

March 2023

1 Opis aplikacji

Nasza aplikacja do analizy różnych cech i charakterystyk klipów audio została napisana w pythonie z użyciem biblioteki wave oraz numpy a także plotly do wizualizacji danych. Strukture projektu można podejrzeć na repozytorium na githubie. Wszystkie funkcje, służace do obliczania charakterystyk sygnału dźwiekowego, znalazły sie w pliku *wave_functions.py*, natomiast ich wykonanie na przykładowym pliku z naszych nagrań znajduje sie w pliku *main.ipynb*. Ustaliliśmy domyślny podział na ramki oraz overlapping ramek na wartości odpowiednio: 10% długości całego pliku i 50% nakładania sie ramek (funkcja *split_to_frames*). W projekcie mamy zarówno funkcje generujace dane wyjściowe, reprezentujace poszczególne charakterystyki na podstawie danych wejściowych w postaci listy wartości w ramkach lub listy wartości z całego pliku audio, jak i funkcje tworzace interaktywne wykresy w pakiecie plotly. Szczegółowe wymagania projektu wymienione sa w pliku *environment.yml*.

2 Opis metod

Implementujac metody, sugerowaliśmy sie opisem zamieszczonym na stronie przedmiotu (*Cechy-sygnału-audio-w-dziedzinie-czasu.pdf*). Podzieliliśmy, zgodnie ze wspomnianym opisem, funkcje na te działajace w zakresie pojedynczej ramki audio i te działajace na całym pliku.

2.1 Cechy sygnału audio na poziomie ramki

2.1.1 Volume

```python
def get_volume(audio,N_):
    return np.sqrt(np.sum(np.power(audio,2))/N_)
```

2.1.2 STE

```
def get_ste(audio, N_):
    return get_volume(audio, N_)**2
```

2.1.3 ZCR

```
def get_zcr(audio,N_):
    return np.sum(np.abs(np.diff(np.sign(audio))))/(2*N_)
```

2.1.4 SR

```
def get_sr(audio, N_, zcr_bound,volume_bound):
    zrc = get_zcr(audio,N_)
    volume= get_volume(audio, N_)
    if zrc > zcr_bound and volume > volume_bound:
        return 1
    else:
        return 0
```

2.1.5 F0

```
def get_f0(audio, l_,amdf=False):
    # amdf - average magnitude difference function
    if amdf:
        return np.sum(np.abs(audio[:-l_] - audio[l_:]))
    else:
        return np.sum(audio[:-l_] * audio[l_:])
```

2.2 Cechy sygnału audio na poziomie klipu

2.2.1 VSTD

```
def get_vstd(audio):
    return np.std(audio)/np.max(np.abs(audio))
```

2.2.2 VDR

```
def get_vdr(audio):
    return (np.max(audio) - np.min(audio))/np.max(audio)
```

2.2.3 VU

```
def get_vu(frames):
    rms = np.sqrt(np.mean(np.square(frames), axis=1))
    rms_db = 20 * np.log10(rms)
    return rms_db
```

2.2.4 LSTER

```
def get_lstr(frame_sec,frame_rate,
↪  percent_frame_size,percent_hop_length):
    frames, n_, N_ = split_to_frames(frame_sec, frame_rate,
    ↪  percent_frame_size,percent_hop_length)
    stes = np.apply_along_axis(get_ste, 1, frames, N_=N_)
    ste_mean = np.mean(stes)
    return np.sum((0.5*ste_mean > stes)+1)/(2*len(frame_sec))
```

2.2.5 Energy Entropy

```
def get_energy_entropy(frames):
    energy = np.sum(np.square(frames), axis=1)
    energy_dist = energy / np.sum(energy)
    return -np.sum(energy_dist * np.log2(energy_dist))
```

2.2.6 ZSTD

```
def get_energy_entropy(frames):
    energy = np.sum(np.square(frames), axis=1)
    energy_dist = energy / np.sum(energy)
    return -np.sum(energy_dist * np.log2(energy_dist))
```

2.2.7 HZCRR

```
def get_hzcrr(frame_sec,frame_rate,
↪  percent_frame_size,percent_hop_length):
    frames, n_, N_ = split_to_frames(frame_sec, frame_rate,
    ↪  percent_frame_size,percent_hop_length)
    zcrs = np.apply_along_axis(get_zcr, 1, frames, N_=N_)
    zcr_mean = np.mean(zcrs)
    return np.sum((1.5*zcr_mean < zcrs)+1)/(2*len(frame_sec))
```

2.3 Spektrogram

Do naszej analizy dołaczyliśmy wizualizacje w postaci spektrogramu, który przedstawia zależność amplitudy i czestotliwości dźwieku od czasu:

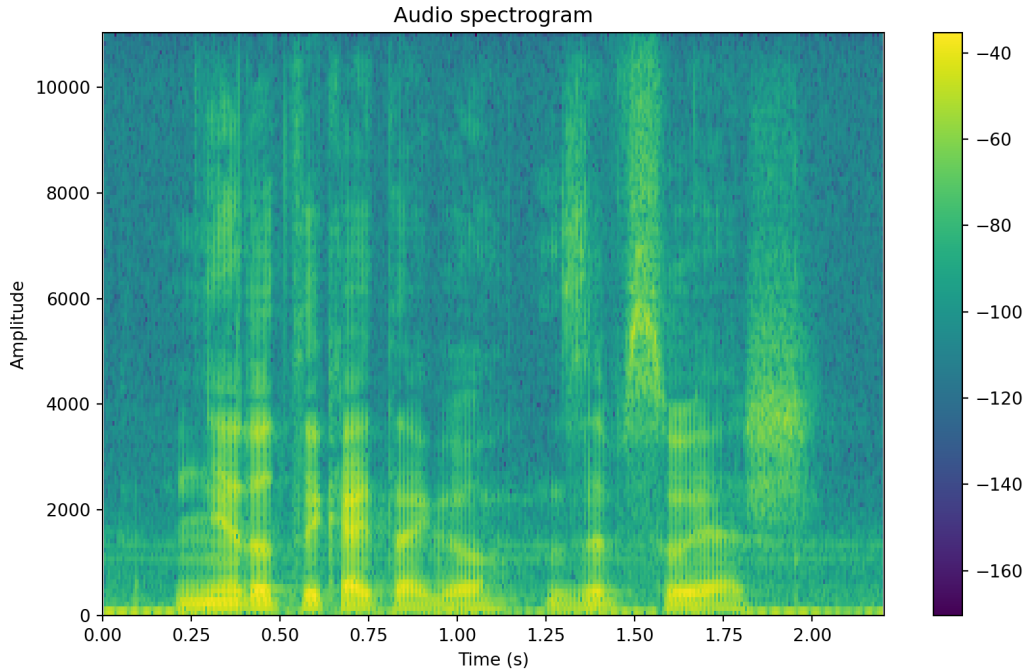

Figure 1: Spektrogram dla jednego z nagrań

3 Wyniki działania

Powyższe funkcje służyły do wygenerowania danych, które nastepnie przedstawiliśmy na interaktywnych wykresach w pakiecie Plotly. Oto przykładowe wizualizacje:

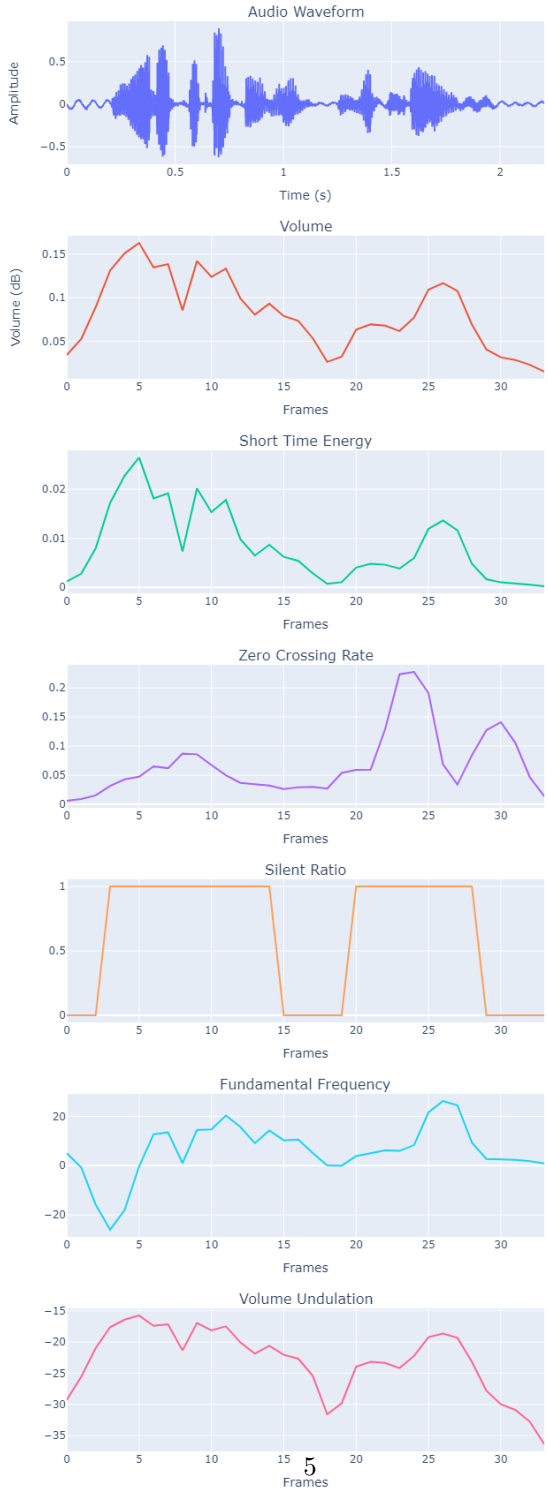

Figure 2: Cechy sygnału na poziomie ramki dla jednego z nagrań

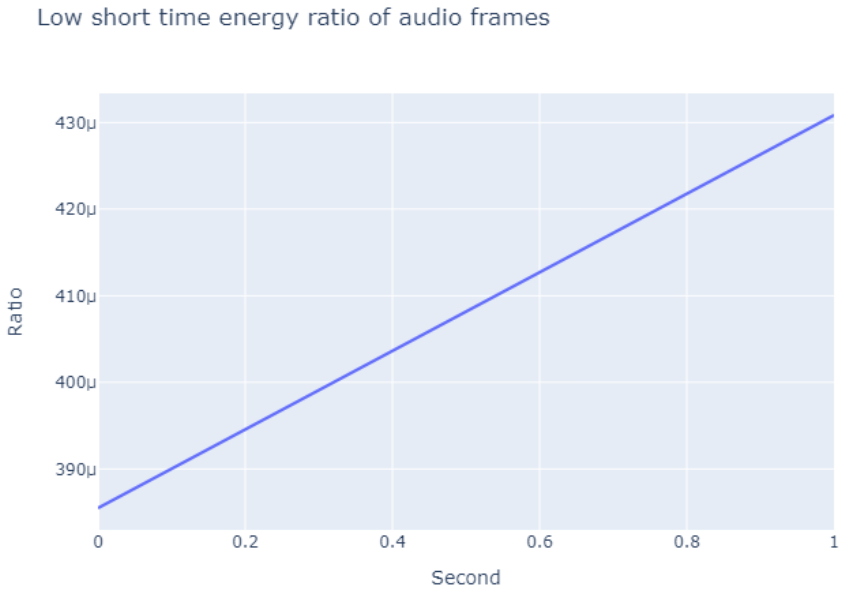

Figure 3: LSTR dla jednego z nagrań

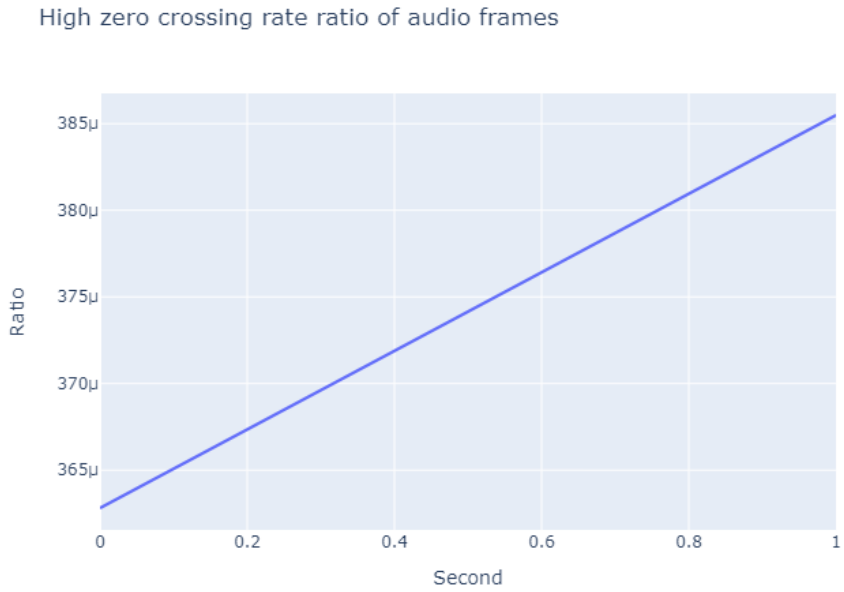

Figure 4: HZCRR dla jednego z nagrań

4 Wnioski z analizy różnych metod i plików

Naszej analizie poddaliśmy kilka plików nagranych na zajeciach, jak też nagranie zawierajace zarówno mowe jak i muzyke, pochodzace z gry komputerowej GTA Vice City. Za interesujace uznaliśmy przede wszystkim wyniki dotyczace Silent Ratio oraz rozróżniania fragmentów mówionych od muzycznych. W wyniku naszej wizualizacji otrzymaliśmy poniższe wykresy:

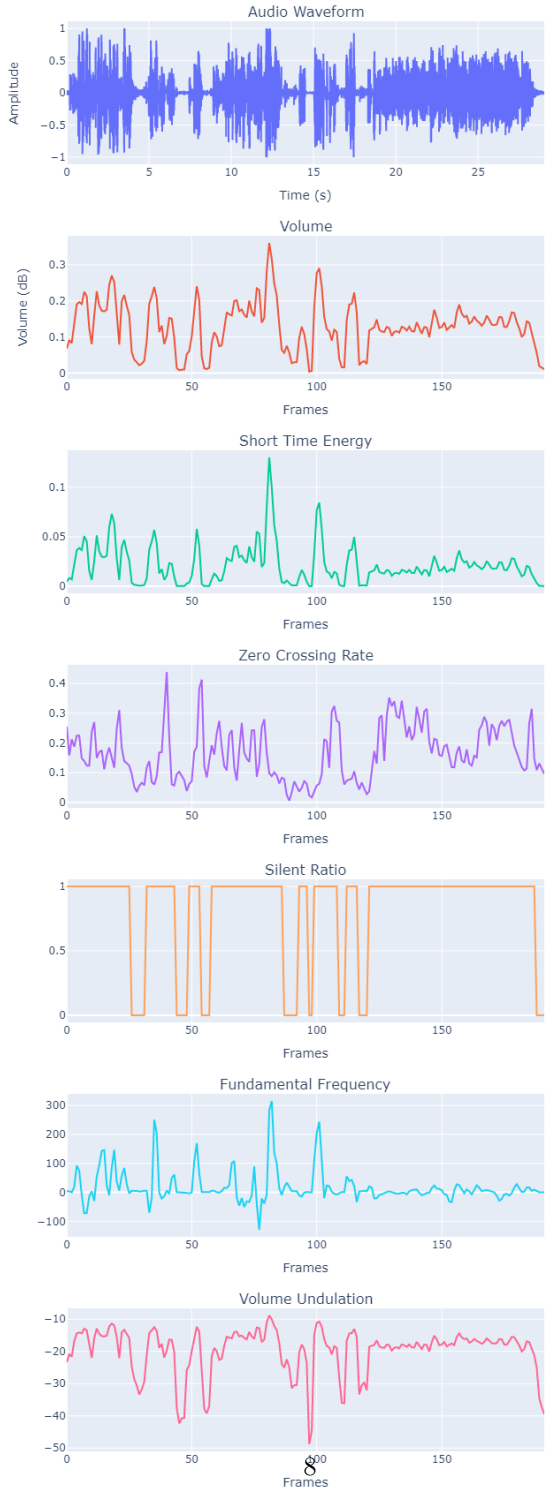

Figure 5: Cechy sygnału na poziomie ramki dla dłuższego klipu

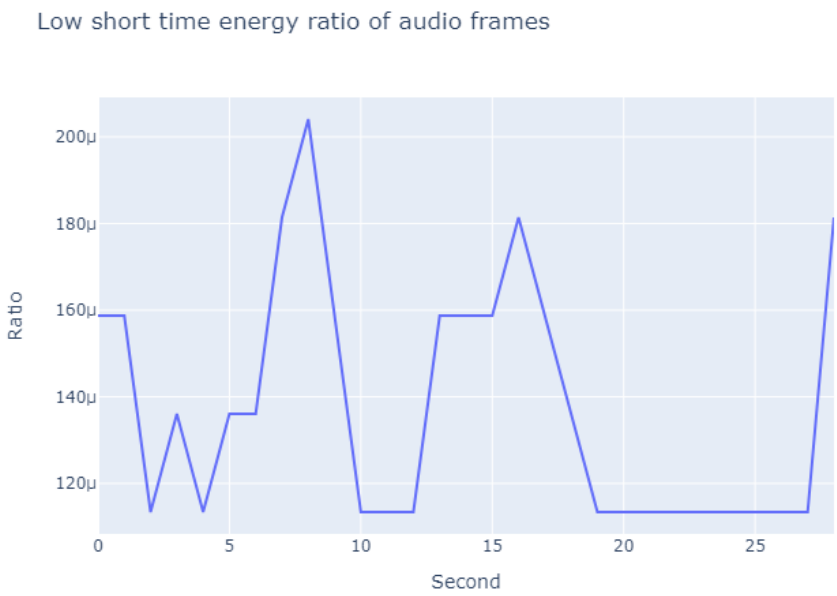

Figure 6: LSTR dla dłuższego klipu

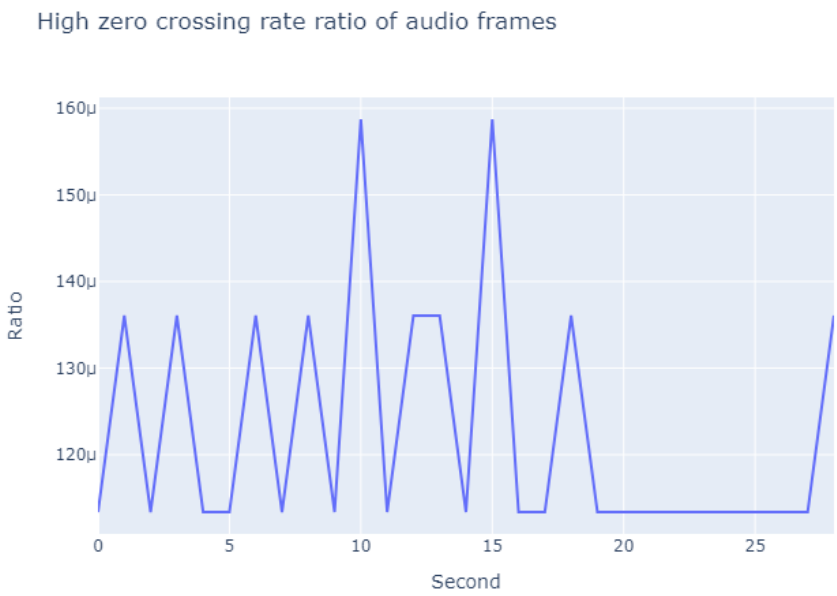

Figure 7: HZCRR dla dłuższego klipu

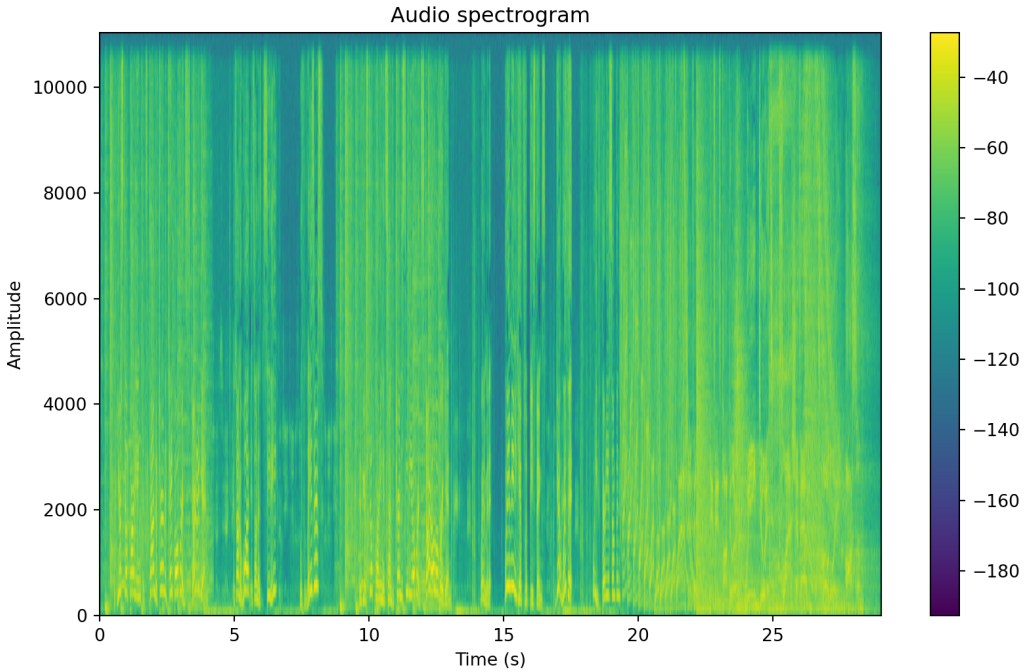

Figure 8: Spektrogram dla dłuższego klipu