Wydział Podstawowych Problemów Techniki
Katedra Optyki i Fotoniki
Kierunek: Optyka
Specjalność: Inżynieria Optyczna

Projekt z Cyfrowego Przetwarzania Sygnału

Tytuł projektu

Rozpoznawanie dźwięków gitary na podstawie częstotliwości fundamentalnej przetwarzanego dźwięku

---

Konrad Markowski

Wrocław 2021

Spis treści

1. Wstęp

Projekt stworzony w języku programowania Python. Program za pomocą algorytmu pyin probabilistycznie określa częstotliwość fundamentalną przetwarzanej próbki dźwiękowej. Uzyskana tablica częstotliwości jest filtrowana z niskich częstotliwości oraz nieokreślonych częstości, a pozostałe wartości są uśredniane w celu określenia częstotliwości dźwięku. Następnie częstotliwość jest porównywana z możliwymi częstotliwościami na gitarze i wybierany zostaje najbliższy dźwięk.

1.1. Założenia

–Standardowy strój gitary - eBGDAE.
–Ograniczenie do progów <0,12>, gdzie próg zerowy to struna 'otwarta'.
–Rozpoznawanie pojedynczego dźwięku.
–Dźwięki zapisane w formacie .wav.
–Częstotliwość próbkowania nagranych dźwięków to 44100 Hz.

1.2. Terminologia/Oznaczenia

Program zawiera 20 próbek, oznaczony według schematu:
Litery e,B,G,D,A,EE/E oznaczają struny, które odpowiadają dźwiękowi 'otwartej' struny, gdzie 'e' oznacza najcieńszą strunę, natomiast 'EE'/'E' - najgrubszą (podwójne 'E' ze względu na konflikt nazw plików dźwiękowych, w innych przypadkach użyte zostanie oznaczenie 'E').

Liczby z zakresu <0,12> oznaczają próg na gitarze, 0 odpowiada dźwiękowi struny 'otwartej', bez przytrzymania żadnego progu gitary.

1.3. Analiza teoretyczna problemu

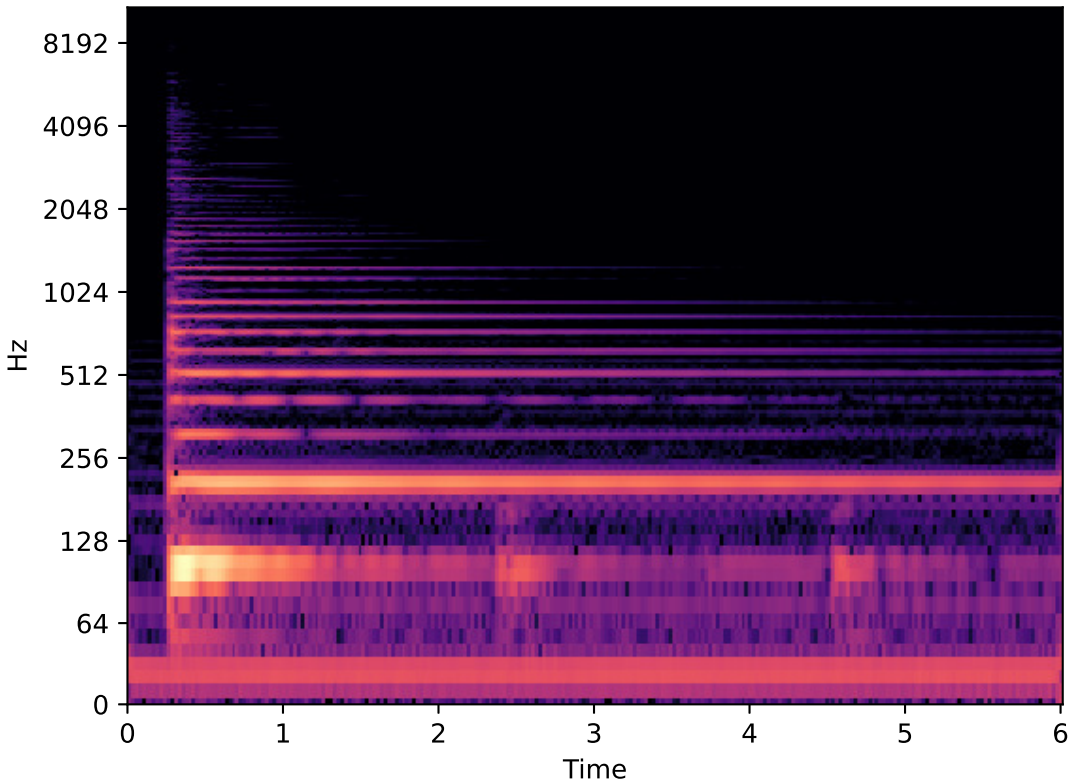
Przetwarzając próbkę dźwiękową za pomocą Transformaty Fouriera można określić dominującą częstotliwość w próbce, która tym samym jest szukaną częstotliwością danego dźwięku gitary.

2. Realizacja

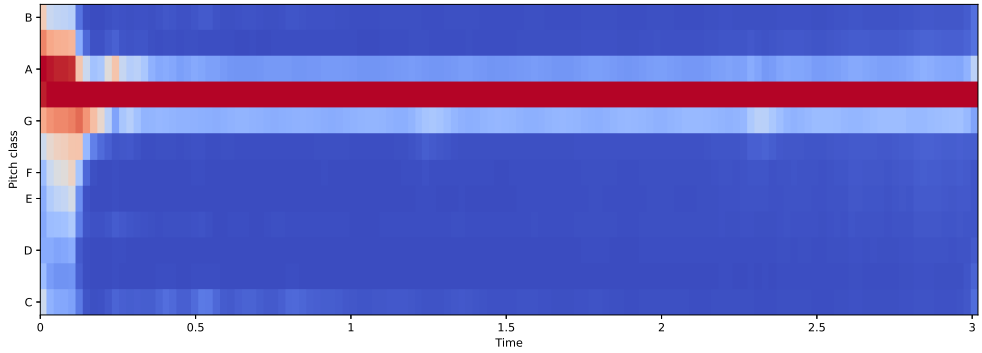
Biblioteka do analizy dźwięku w Pythonie - librosa pozwala na łatwe wczytanie pliku dźwiękowego oraz oferuje wiele funkcji do jego analizy. W kontekście projektu, dwie funkcje STFT (Short term Fourier Transform) oraz CQT (constant-Q transform) realizują najlepiej podejście teoretyczne jednak zostały odrzucone na rzecz funkcji PYIN. Głównym powodem była trudność w ich obsłudze.

W poniższych przykładach użytym plikiem dźwiękowym był dźwięk D6 (plik D6.wav).

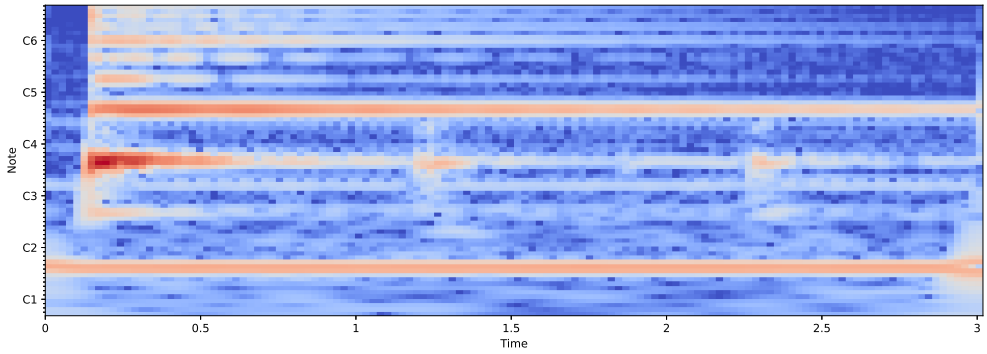
Widmo otrzymane z wykorzystaniem STFT:



Chromatogram z STFT:



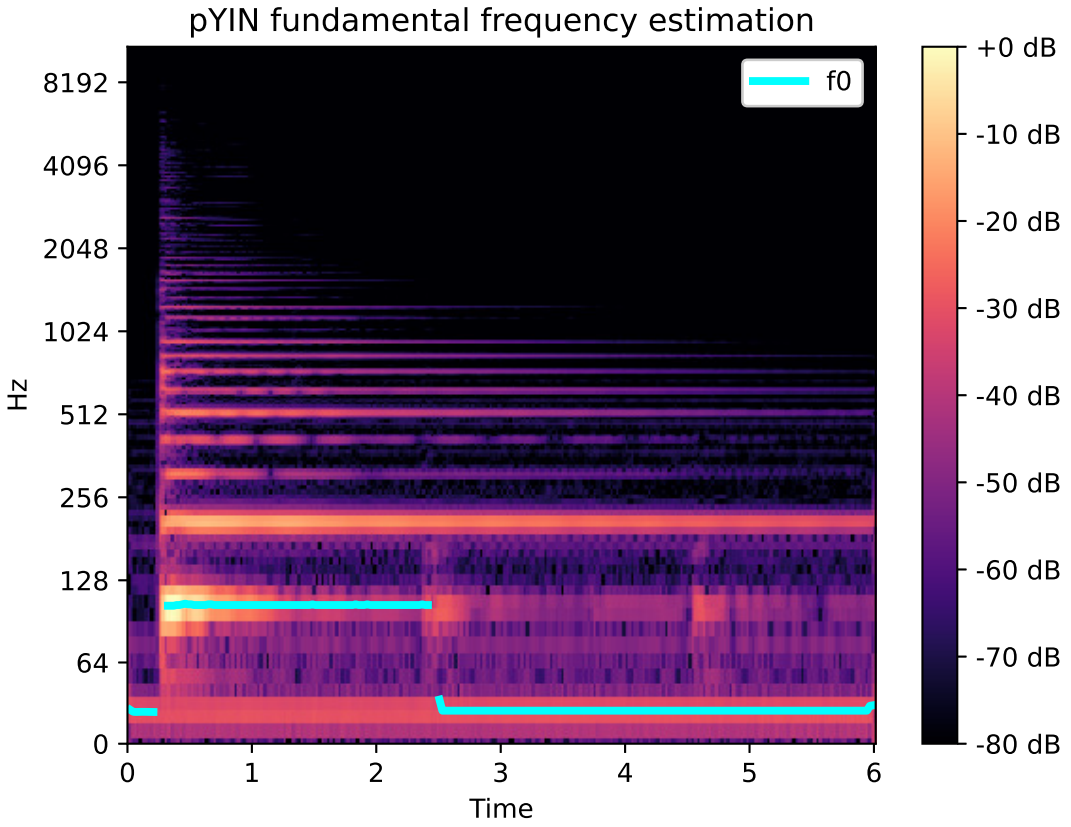
Widmo z CQT:



W obu metodach, co widać w szczególności na chromatogramie z STFT można otrzymać szukaną częstotliwość na wykresie. Problemem było wyciągnięcie wartości częstotliwości w postaci liczbowej, co nie udało mi się w żadnej z metod. W przypadku wykorzystania CQT, możliwe jest odczytanie wartości maksymalnej, która odpowiada szukanej częstotliwości, jednak nie udało odszukać mi się sposobu na przeliczenie tej wartości na częstotliwość.

Funkcja PYIN, wykorzystuje algorytm PYIN który za pomocą probabilistycznego poziomu rozkładu(probilistic treshold distributions) określa częstotliwość próbki. Zaletą tej funkcji jest bezpośredni dostęp do tablicy zawierającą wartości uzyskanych częstotliwości. Minusem jest długi czas przetwarzania próbki przez funkcję oraz trudność w określeniu częstotliwości minimalnych i maksymalnych, które wpływają na wyniki.

Wizualizacja działania funkcji PYIN:



2.1. Zawartość programu

Program składa się z 3 plików z kodem:

main.py - uruchomienie funkcji main rozpoczyna analizę próbek znajdujących się w folderze 'sounds'.

SoundRecognition.py - zawierający klasę SoundRecognition odpowiedzialną za rozpoznanie dźwięku, a także określenie poprawnego dźwięku (metoda recognizeSound) na podstawie nazwy pliku (metoda pathNameToNote).

SoundTable.py - zawierający klasę GuitarStrings, która generuję listę słowników, gdzie każdy ze słowników reprezentuje jedną strunę.

Oprócz tego w folderze 'sounds' znajduje się 20 plików dźwiękowych w formacie .wav oznaczonych zgodnie z wcześniej przyjętą konwencją. Pliki własne, przycięte oraz przepuszczone przez filtr high-pass dla 40Hz, w celu usunięcia szumów (nagranie i obrobienie dźwięków w programie Audacity).

2.2. Działanie programu

Uruchomienie funkcji main z pliku main.py powoduje wywołanie funkji getFileNames, która zbiera nazwy plików znajdujące się w folderze 'sounds'.Tym samym możliwe jest dodanie kolejnych próbek, jednak żeby metoda pathNameToNote poprawnie zadziałała, wymagane jest oznaczenie pliku według wcześniej przyjętej konwencji.
W pętli wywoływane są metody pathNameToNote oraz recognizeSound. Następnie w oknie konsoli wyświetlane są po sobie wyniki (formatowanie wyświetlania wyników nie jest zrobione dokładnie - wyniki się rozjeżdżaj względem innych rzędów). Na sam koniec wyświetlany jest współczynnik poprawności w [%].
Program w zależności od parametrów fmin i fmax w funkcji pyin daje różne rezultaty, w szczególności parametr fmin ma decydujący wpływ na wynik, a także na czas wykonywania programu (mniejsze fmin -> wolniejszy program). fmin został ustawiony na 30Hz dla których wyniki są zadowalające oraz czas wykonywania programu jest znośny. Warto dodać, że rekomendowaną częstotliwością minimalną jest około 76Hz, natomiast najniższą częstotliwością dla gitary jest około 41Hz dla E0.

2.3. Uwagi

Program wymaga następujących bibliotek:
–matplotlib,
–numpy,
–librosa.

Klasa GuitarStrings w pliku SoundTable.py została stworzona z myślą o wykorzystaniu jej również do określenia nie tylko dźwięku, a także struny. Mimo że częstotliwości dla dwóch sąsiednich strun nakładają się w pewnym momencie i wytwarzane dźwięki mają taką samą częstotliwość (np. e0 i B5), to długość strun jest inna, a więc ton dźwięku jest inny, przez co prawdopodobnie możliwe jest znalezienie różnic pomiędzy dźwiękami oraz ich rozróżnienie. Ze względu na stopień skomplikowania problemu rozpoznawanie strun zostało zaniechane.

Ponadto zapisane wartości częstotliwości są dwa razy większe od rzeczywistych lub innymi słowy, są oktawę wyżej. Wynika to z przyjętej konwencji zapisu nut gitary oktawę

wyżej na pięciolinii. A więc żeby zachować konwencję również częstotliwości są oktawę wyżej, co jest dosyć mylące ale ze względu na fakt, że podczas przeszukiwania internetu w poszukiwaniu informacji, nie spotkałem się z innymi sposobem zapisu, również zdecydowałem się taki przyjąć.

Jednocześnie kolumna TrueFreq, w oknie konsoli przy wywołaniu funkcji main z pliku main.py, pokazuje częstotliwość obliczoną przez algorytm pyin, która nie jest zapisana w tej konwencji, to znaczy, uzyskana wartość jest rzeczywistą wartością, jeżeli przyjmiemy, że program działa poprawnie.

3. Wnioski

Analiza sygnału jest dużo bardziej skomplikowana w rzeczywistych warunkach, z rzeczywistym sygnałem. Ponadto narzędzia stworzone do analizy sygnału dźwiękowego (w tym przypadku biblioteka librosa) wymaga wiedzy z zakresu działania tych funkcji, żeby umiejętnie z nich korzystać.

Program jest w stanie rozpoznać dźwięk z dużą skutecznością, zatem można określić projekt sukcesem biorąc pod uwagę jak skomplikowanym zagadnieniem jest rozpoznawanie dźwięku oraz jak bardzo udało mi się uprościć problem. Współczesne programy do rozpoznawania dźwięków bazują na kliku systemach rozpoznawania, które wspólnie określają dźwięk, a nawet rozpoznają struny. Wykorzystanie uczenia maszynowego prawdopodobnie jest najlepszym podejściem do problemu, jeżeli ma się dostęp do wielu próbek.

4. Dodatek

Tabele dźwięków gitary dla standardowego stroju gitary eBGDAE. Dane do tabeli uzyskane z metody print_table klasy GuitarStrings (plik SoundTable.py). Wartości liczbowe w Hz.

Tabela z dźwiękami:

	0	1	2	3	4	5	6	7	8	9	10	11	12
e:	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E
B:	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
G:	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G
D:	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D
A:	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A
E:	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E

Tabela z 'przyjętymi' częstotliwościami (oktawę wyżej):

	0	1	2	3	4	5	6	7	8	9	10	11	12
e:	330	349	370	392	415	440	466	494	523	554	587	622	660
B:	247	261	277	293	311	329	349	370	392	415	440	466	494
G:	196	207	220	233	246	261	277	293	311	329	349	369	392
D:	147	155	165	174	185	196	207	220	233	247	261	277	294
A:	110	116	123	130	138	146	155	164	174	184	195	207	220
E:	82	86	92	97	103	109	115	122	130	137	146	154	164

Tabela z rzeczywistymi częstotliwościami:

	0	1	2	3	4	5	6	7	8	9	10	11	12
e:	165	174.5	185	196	207.5	220	233	247	261.5	277	293.5	311	330
B:	123.5	130.5	138.5	146.5	155.5	164.5	174.5	185	196	207.5	220	233	247
G:	98	103.5	110	116.5	123	130.5	138.5	146.5	155.5	164.5	174.5	184.5	196
D:	73.5	77.5	82.5	87	92.5	98	103.5	110	116.5	123.5	130.5	138.5	147
A:	55	58	61.5	65	69	73	77.5	82	87	92	97.5	103.5	110
E:	41	43	46	48.5	51.5	54.5	57.5	61	65	68.5	73	77	82

5. Bibliografia

[1] https://www.eecs.qmul.ac.uk/ simond/pub/2014/mauchdixon-pyin-icassp2014.pdf.

[2] http://librosa.org/doc/main/generated/librosa.stft.html.

[3] https://librosa.org/doc/main/generated/librosa.load.html.

[4] https://myguitarpal.com/the-guitar-sounds-one-octave-lower-than-written/.

[5] https://stackoverflow.com/questions/43877971/librosa-pitch-tracking-stft.

[6] https://librosa.org/doc/main/generated/librosa.pyin.html.

[7] https://stackoverflow.com/questions/68463262/getting-frequencies-by-fft-frequencies-function-in-librosa.

[8] https://stackoverflow.com/questions/63350459/getting-the-frequencies-associated-with-stft-in-librosa/.

[9] https://www.quora.com/whats-the-difference-between-notes-in-guitar-on-different-strings.

[10] https://pages.mtu.edu/ suits/notefreqcalcs.html.