AKADEMIA GÓRNICZO-HUTNICZA

im. Stanisława Staszica w Krakowie

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

Praca dyplomowa
inżynierska

Nina Utrata
Imię i nazwisko

Inżynieria akustyczna
Kierunek studiów

Opracowanie metody segmentacji wielonutowych próbek instrumentów dętych
Temat pracy dyplomowej

dr Marek Pluta ……..
Promotor pracy *Ocena*

Kraków, rok 2018/2019

Kraków, 22.01.2019 r.

Imię i nazwisko: Nina Utrata
Nr albumu: -
Kierunek studiów: Inżynieria akustyczna
Profil dyplomowania: -

OŚWIADCZENIE

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tj. Dz.U.z 2006 r. Nr 90, poz. 631 z późn.zm.) : „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust.1 ustawy z dnia 27 lip[ca 2005 r. Prawo o szkolnictwie wyższym (tj. Dz.U. z 2012 r. poz. 572, z późn.zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem (-am) ze źródeł innych niż wymienione w pracy".

.....
podpis dyplomanta

Kraków, 22.01.2019 r.

Imię i nazwisko: Nina Utrata

Nr albumu: -

Kierunek studiów: **Inżynieria akustyczna**

Profil dyplomowania: -

OŚWIADCZENIE

Świadoma odpowiedzialności karnej za poświadczanie nieprawdy oświadczam, że niniejszą inżynierską pracę dyplomową wykonałem/łam osobiście i samodzielnie oraz nie korzystałem/łam ze źródeł innych niż wymienione w pracy.

Jednocześnie oświadczam, że dokumentacja oraz praca nie narusza praw autorskich
w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r. Nr 90 poz. 631 z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym. Nie zawiera ona również danych i informacji, które uzyskałam w sposób niedozwolony. Wersja dokumentacji dołączona przeze mnie na nośniku elektronicznym jest w pełni zgodna z wydrukiem przedstawionym do recenzji.

Zaświadczam także, że niniejsza inżynierska praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

……..
podpis dyplomanta

Kraków, 22.01.2019 r.

Imię i nazwisko:	Nina Utrata
Adres korespondencyjny:	-
Temat pracy dyplomowej inżynierskiej:	Opracowanie metody segmentacji wielonutowych próbek instrumentów dętych
Rok ukończenia:	2019
Nr albumu:	-
Kierunek studiów:	Inżynieria akustyczna
Profil dyplomowania:	-

OŚWIADCZENIE

Niniejszym oświadczam, że zachowując moje prawa autorskie , udzielam Akademii Górniczo-Hutniczej im. S. Staszica w Krakowie nieograniczonej w czasie nieodpłatnej licencji niewyłącznej do korzystania z przedstawionej dokumentacji inżynierskiej pracy dyplomowej, w zakresie publicznego udostępniania i rozpowszechniania w wersji drukowanej i elektronicznej[1].

Publikacja ta może nastąpić po ewentualnym zgłoszeniu do ochrony prawnej wynalazków, wzorów użytkowych, wzorów przemysłowych będących wynikiem pracy inżynierskiej[2].

Kraków, ...… ……..
data *podpis dyplomanta*

[1] Na podstawie Ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (Dz.U. 2005 nr 164 poz. 1365) Art. 239. oraz Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. z 2000 r. Nr 80, poz. 904, z późn. zm.) Art. 15a. "Uczelni w rozumieniu przepisów o szkolnictwie wyższym przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli uczelnia nie opublikowała pracy dyplomowej w ciągu 6 miesięcy od jej obrony, student, który ją przygotował, może ją opublikować, chyba że praca dyplomowa jest częścią utworu zbiorowego."

[2] Ustawa z dnia 30 czerwca 2000r. – Prawo własności przemysłowej (Dz.U. z 2003r. Nr 119, poz. 1117 z późniejszymi zmianami) a także rozporządzenie Prezesa Rady Ministrów z dnia 17 września 2001r. w sprawie dokonywania i rozpatrywania zgłoszeń wynalazków i wzorów użytkowych (Dz.U. nr 102 poz. 1119 oraz z 2005r. Nr 109, poz. 910).

Kraków, dnia 22.01.2019 r.

AKADEMIA GÓRNICZO-HUTNICZA
WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

TEMATYKA PRACY DYPLOMOWEJ INŻYNIERSKIEJ
dla studenta IV roku studiów stacjonarnych

Nina Utrata
imię i nazwisko studenta

TEMAT PRACY DYPLOMOWEJ INŻYNIERSKIEJ:
Opracowanie metody segmentacji wielonutowych próbek instrumentów dętych

Promotor pracy: dr Marek Pluta

Recenzent pracy: dr hab. inż. Mariusz Giergiel, prof. AGH *Podpis dziekana:*

PLAN PRACY DYPLOMOWEJ :
1. Omówienie tematu pracy i sposobu realizacji z promotorem.
2. Udostępnienie przez promotora nagrań instrumentów muzycznych potrzebnych do pracy.
3. Zebranie i opracowanie literatury dotyczącej tematu pracy.
4. Zebranie i opracowanie wyników badań.
5. Analiza wyników badań, ich omówienie i zatwierdzenie przez promotora.
6. Opracowanie redakcyjne.

Kraków, ...… ……..
data *podpis dyplomanta*

TERMIN ODDANIA DO DZIEKANATU: ..... 20..... r.

podpis promotora

Akademia Górniczo-Hutnicza im. Stanisława Staszica Kraków, 22.01.2019 r.
Wydział Inżynierii Mechanicznej i Robotyki

Kierunek: Inżynieria akustyczna

Profil dyplomowania: -

Nina Utrata
Praca dyplomowa inżynierska
Opracowanie metody segmentacji wielonutowych próbek instrumentów dętych
Opiekun: dr Marek Pluta

STRESZCZENIE

W pracy przedstawiono propozycję rozwiązania zagadnienia jakim jest stworzenie aplikacji służącej do automatycznej segmentacji wielonutowych próbek instrumentów dętych. Na początku określono trudności związane z tym zagadnieniem, następnie przedstawiono rozwiązanie przy pomocy technik maszynowego uczenia oraz przetwarzania sygnałów.

Pierwsza część pracy skupia się na przybliżeniu czytelnikowi wiedzy teoretycznej potrzebnej do zrozumienia istoty działania algorytmów. Opisane tam zostały współczynniki MFCC (ang. *Mel Frequency Cepstral Coefficient*), modele GMM (ang. *Gaussian Mixture Model*) oraz niezbędna terminologia muzyczna. Jako, że niniejsza praca wchodzi w skład MIR (ang. *Music Information Retrieval*) czyli dziedziny łączącej w sobie nauki muzykologii, akustyki oraz programowania odniesiono się do niej oraz przybliżono istniejące obecnie biblioteki wchodzące w jej skład.

W drugiej części pracy omówiono obsługę aplikacji oraz przybliżono jej techniczną budowę. Na koniec przeprowadzono testy sprawdzające skuteczność algorytmów oraz porównano wyniki analizy z wynikami uzyskanymi przy pomocy istniejących narzędzi.

AGH University of Science and Technology Kraków, 22.01.2019r.
Faculty of Mechanical Engineering and Robotics

Field of Study: Acoustic engineering

Specialisations: -

Nina Utrata
Engineer Diploma Thesis
Development of the segmentation method for multi-pitch wind instrument
Supervisor: dr Marek Pluta

SUMMARY

This paper presents the proposition of solution in the topic of the application for automatic segmentation method for multi-pitch wind instrument samples. Firstly, the difficulties were determined that may occur in the selected subject. Then the solution was presented with the machine learning and sound processing methods.

First part of this thesis focuses on introducing theoretical knowledge to the reader that is needed to understand how the algorithms work. It describes the issues of Mel Frequency Cepstral Coefficient, Gaussian Mixture Models and necessary music terminology. This paper is related to the Music Information Retrieval involving musicology, acoustics and programming therefore it was necessary to explain the discipline and present already existing programming packages that work in this field .

In the other half of the paper the operating manual of the application was explained and also the technical part of the program was described. Finally, the effectiveness of the application was tested. The final results from the application described in this paper were compared with analysis made with already existing tools.

Spis treści

1. Wstęp

Stale zwiększająca się moc obliczeniowa współczesnych komputerów i superkomputerów, zwłaszcza na przestrzeni ostatnich dwóch dekad, doprowadziła do dynamicznego rozwoju uczenia maszynowego - czyli dziedziny wchodzącej w skład nauk zajmujących się problematyką sztucznej inteligencji. Jej głównym celem jest tworzenie algorytmów, które w sposób automatyczny potrafią pozyskać oraz przeanalizować dane, zwiększając tym samym ich skuteczność w rozwiązywaniu problemów.

Uczenie maszynowe znajduje zastosowanie między innymi w dziedzinach opierających się na wyszukiwaniu i klasyfikowaniu wzorców, w których algorytm początkowo przetwarza i tworzy modele na podstawie dostarczonych danych „uczących", a następnie jest w stanie dopasować do nich nieznane wcześniej dane „testowe" wprowadzone przez użytkownika.

Jednym z ciekawszych zagadnień związanym z analizą muzyki w kontekście niniejszej pracy jest MIR (ang. *Music Information Retrieval*), który łączy przetwarzanie sygnałów z muzykologią, akustyką oraz programowaniem. Opiera się na uczeniu maszynowym przy użyciu narzędzi związanych z analizą częstotliwości i dynamiki sygnałów (między innymi FFT, spektrogramy oraz MFCC (ang. *Mel Frequency Cepstral Coefficients*)), pobierając z utworów informacje o ich rytmie, nastroju, gatunku muzycznym, czy też tonacji. Tego typu automatyczna analiza plików dźwiękowych używana jest przez wiele obecnych na rynku aplikacji audio oraz mediów strumieniowych.

Dobrym przykładem wykorzystania technologii wchodzących w skład Music Information Retrieval jest popularna obecnie aplikacja „Shazam", która umożliwia rozpoznanie tytułu piosenki oraz jej wykonawcy na podstawie nagranego z telefonu fragmentu utworu. Jest to możliwe dzięki obszernej bazie danych, w której przechowywanych jest około 8 milionów „śladów" piosenek. Powstają one w oparciu o algorytmy wykorzystujące spektrogramy i techniki wyszukujące w nich charakterystyczne cechy [1]. Jak się okazuje, tego typu parametryzacja sygnału audio jest deterministyczna, dzięki czemu przeprowadzenie tej samej analizy nawet na małym fragmencie nieznanej piosenki przesłanym przez użytkownika jest wystarczające, aby aplikacja była w stanie odnaleźć oryginalny utwór w bazie.

Niniejsza praca skupia się na pozyskiwaniu informacji odnośnie występujących w danym utworze wysokości dźwięków przy wykorzystaniu wyżej opisanych technologii.

Streszczenie rozdziałów

W pierwszym rozdziale określony jest cel pracy oraz zakres działania aplikacji. Skupiono się tam również na spodziewanych trudnościach związanych z podjętą tematyką pracy.

Drugi rozdział poświęcony jest wyjaśnieniu terminów, które są niezbędne do zrozumienia działania algorytmów (MFCC, GMM), ponadto wyjaśniono w nim terminologię muzyczną niezbędną do zrozumienia istoty działania programu.

W rozdziale trzecim zaprezentowano istniejące obecnie na rynku biblioteki zajmujące się pojęciami z dziedziny MIR wraz z wyszczególnieniem konkretnych funkcji, które mogą być wykorzystane przy porównaniu wyników z tych bibliotek do efektów działania aplikacji opisanej w pracy.

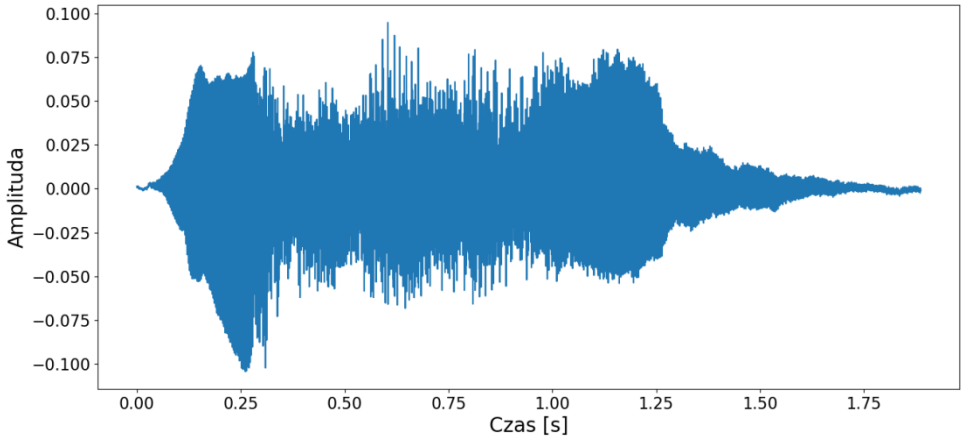
Czwarty rozdział opisuje techniczną budowę programu. Zamieszczono w nim teoretyczny plan działania algorytmów, opis interfejsu aplikacji, wymieniono użyte biblioteki, dołączono diagram klas wraz z rozwinięciem funkcji danych obiektów klas.

W piątym rozdziale zawarta jest analiza wyników testów prawidłowości działania aplikacji.

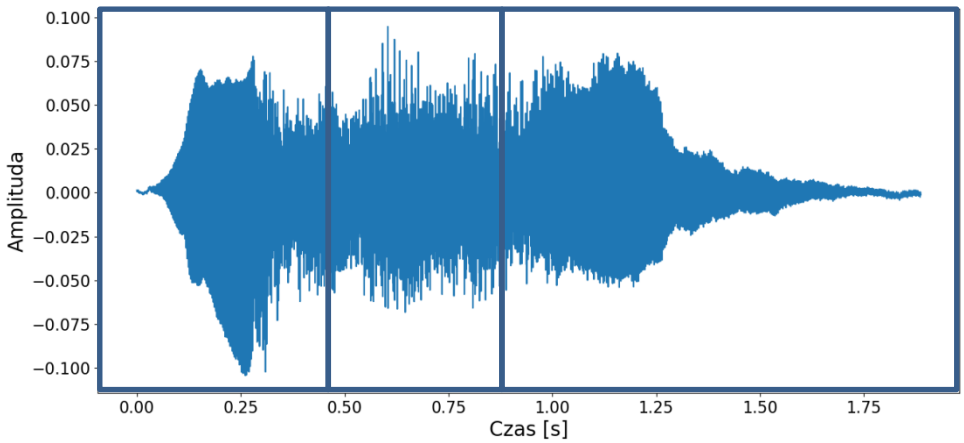
Rozdział szósty podsumowuje realizację pracy i opisuje końcowe wnioski.

1.1 Cel i zakres pracy

Głównym celem niniejszej pracy jest stworzenie aplikacji napisanej w języku programowania Python, **służącej do automatycznej segmentacji fragmentów utworu muzycznego ze względu na występujące w nim wysokości dźwięków**. Wynikiem działania algorytmu są podzielone części nagrania, które następnie mogą zostać zapisane na dysku z nazwą odnoszącą się do rozpoznanej wysokości. Poniżej przedstawiona jest prezentacja efektu jaki uzyskamy poprzez analizę próbki.



Rys. 1.1. Przykładowa graficzna reprezentacja sygnału audio, którego rozkład wysokości nie jest znany



Rys. 1.2. Efekt automatycznej segmentacji, gdzie każdy z zaznaczonych obszarów obejmuje fragment dźwięku o pewnej, rozpoznanej wysokości

Użytkownik programu ma do wyboru jeden z trzech instrumentów dętych: flet, obój lub klarnet, którego nagrania może analizować. Aby analiza sygnału była poprawna, próbki, które mają zostać poddane segmentacji muszą być solowymi nagraniami jednego instrumentu. Ma to związek z faktem, że wykorzystane w tej pracy algorytmy korzystają z dedykowanych modeli wysokości dźwięków w zależności od wybranego instrumentu. W rezultacie ta sama wysokość dźwięku (np. $a^1 = 440Hz$) grana na flecie będzie analizowana na podstawie innej bazy próbek, niż grana na klarnecie. W sytuacji, w której użytkownik zaznaczy, że program ma analizować nagrania fletu, a wskaże próbki klarnetu, program nie zadziała poprawnie.

Drugim zadaniem, po ukończeniu prac związanych z tworzeniem aplikacji, jest przeprowadzenie testów sprawdzających, z jaką skutecznością program działa. W tym celu utworzono bazę plików, których zawartość jest znana by następnie poddać je analizie. W efekcie sprawdzając ile z nich zostało w pełni dobrze rozpoznanych, a w przypadku złego rozpoznania co wpłynęło na błędne końcowe wyniki.

1.2 Trudności związane z analizą nagrań utworów muzycznych

Przed rozpoczęciem działań związanych z budową aplikacji istotne jest określenie problemów mogących wystąpić przy jej realizacji. Przy analizie nagrań instrumentów muzycznych pod względem występujących w nich wysokości oczywistym sposobem wydaję się analiza częstotliwościowa. Obliczenie widma lub spektrogramu z sygnału akustycznego dostarcza cennych informacji odnośnie rozkładu amplitudowo-częstotliwościowego. Większość dostępnych na rynku aplikacji wykorzystuje te operacje do wyznaczenia tonu podstawowego by następnie dopasować wynik do odpowiadającego mu dźwięku ze skali muzycznej.

Wyzwaniem jest sprawienie, by komputer był w stanie rozpoznać na jaką wysokość wskazuje dany rozkład harmonicznych oraz w którym momencie jeden dźwięk przechodzi w drugi, w sytuacji gdzie poddany analizie sygnał jest wielonutowym nagraniem, a nie pojedynczym dźwiękiem. Do rozwiązania tego zagadnienia nieprzydatna jest dynamika sygnału, gdyż tempo grania utworu cechuje się zmiennością, poszczególne dźwięki mają różną długość trwania oraz mogą być grane w wielu stopniach natężenia. Sama analiza spektrogramów również nie determinuje w jaki sposób powinna przebiegać segmentacja, gdyż występuje zjawisko nakładania się kolejnych dźwięków na siebie, w taki sposób, że gdy jedna wysokość zaczyna grać poprzednia jeszcze wybrzmiewa w tle, co ma wpływ na widmo sygnału i utrudnia rozpoznanie tonu podstawowego w krótkiej chwili czasowej nagrania.

Algorytm powinien być w stanie określić rozkład wysokości w utworze niezależnie od powyższych aspektów.

Biorąc pod uwagę wymienione trudności, w niniejszej pracy wysokości nagrań nie są definiowane tylko na podstawie analizy widma lub obrazu spektrogramu. Modele dźwięków dla poszczególnych instrumentów są tworzone przy pomocy algorytmów wykorzystujących techniki maszynowego uczenia, gdzie jako dane uczące przyjmujemy zdefiniowane nagrania wysokości, które posłużą za wzorzec przy analizie.

W ten sposób unika się trudności związanych z ręcznym dopasowywaniem reguł mających służyć segmentacji. Ponadto dodatkowym atutem takiego rozwiązania jest możliwość prostego poszerzania biblioteki analizowanych instrumentów, gdyż algorytm działa w ten sam sposób niezależnie od instrumentu.

2. Wprowadzenie teoretyczne

2.1 Terminologia muzyczna

Psychoakustyczna definicja określa wysokość dźwięku jako atrybut wrażenia słuchowego, pozwalający uszeregować dźwięki od niskich do wysokich na skali wykorzystywanej w muzyce [2]. W terminologii muzycznej o wysokości dźwięku decyduje wyłącznie częstotliwość (w przypadku tonów) lub częstotliwość podstawowa (w przypadku wielotonów harmonicznych) [2, 3, 4].

Ton – sygnał mający sinusoidalny przebieg o ściśle określonej częstotliwości, amplitudzie i fazie.

Wielotony harmoniczne – sygnał składający się z sumy tonów o ściśle określonej częstotliwości podstawowej, amplitudzie i fazie [3] :

$$u(t) = \sum_{n=1}^{N} A_n \cos(n\omega t + \varphi_n) \tag{2.1}$$

$$\omega = \frac{2\pi}{T} = 2\pi f \ \ [rad/s] \tag{2.2}$$

gdzie:

N – całkowita liczba harmonicznych wielotonu (jeśli N=1 to wzór odnosi się do tonu),

A_n - amplituda sygnału,

ω - częstotliwość kątowa [rad⁄s],

φ_n – przesunięcie fazowe [rad],

f – częstotliwość (dla tonu) / częstotliwość podstawowa (dla wielotonu harmonicznego) [Hz].

Zbiór wszystkich dostępnych wysokości muzycznych wyznacza system dźwiękowy. W pracach naukowych badających zjawiska muzyczne zazwyczaj wykorzystywany jest system równomiernie temperowany. Określa on jednoznacznie kolejne dźwięki skali muzycznej za pomocą stosunków ich częstotliwości, dzięki czemu

umożliwia obliczenia oparte na fizycznych parametrach dźwięku. Najczęściej jako strój tego systemu przyjmuje się a^1=440Hz.

Skala wysokości muzycznych jest okresowa, a jej okresem jest oktawa, która dzielona jest w tym systemie na dwanaście półtonów. W ramach jednej oktawy wyróżnia się siedem podstawowych nazw wysokości, wyliczając od najniższej: c,d,e,f,g,a,h, gdzie między poszczególnymi dźwiękami występuje odległość dwóch półtonów (czyli całego tonu) lub jednego półtonu (Rys. 2.1). Te same nazwy dźwięku w sąsiednich oktawach są ze sobą powiązane zależnością, że stosunek ich częstotliwość wynosi 2 [4].

$$Półton: \frac{f_2}{f_1} = \sqrt[12]{2} \approx 1{,}059 \tag{2.3}$$

$$Cały\ ton: \frac{f_2}{f_1} = \sqrt[6]{2} \approx 1{,}122 \tag{2.4}$$

gdzie:

f_1, f_2 – częstotliwości dwóch wysokości dźwięków, gdzie f_2 jest wyższym dźwiękiem niż f_1 [Hz]

c	*d*	*e*	*f*	*g*	*a*	*h*	*c*
	Cały ton	Cały ton	Półton	Cały ton	Cały ton	Cały ton	Półton

Rys. 2.1 Schemat rozkładu wysokości w ramach oktawy

Tab. 2.1. Przykład rozkładu wysokości dźwięków w oktawach wraz z odpowiadającą im częstotliwością podstawową, w systemie równomiernie temperowanym o stroju a^1=440Hz

Nazwa oktawy	Nazwa dźwięku i odpowiadająca mu częstotliwość [Hz]						
Razkreślna	c^1	d^1	e^1	f^1	g^1	a^1	h^1
	261,6	293,7	329,6	349,2	391,9	440,0	493,9
Dwukreślna	c^2	d^2	e^2	f^2	g^2	a^2	h^2
	523,3	587,3	659,3	698,5	784,0	880,0	987,8

W niniejszej pracy termin „wysokość dźwięku” oznacza nazwę dźwięku w konkretnej oktawie przy wykorzystaniu systemu równomiernie temperowanego o stroju a^1=440Hz.

2.2 MFCC (ang. *Mel Frequency Cepstral Coefficients*)

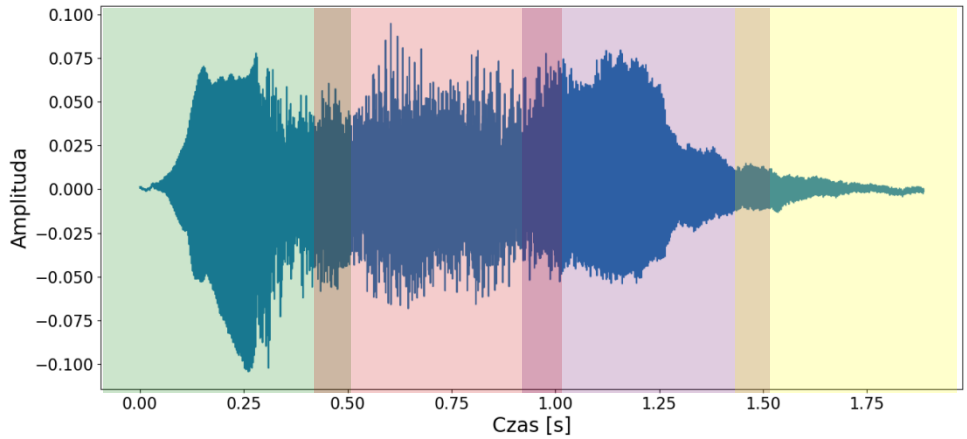
Skala melowa - skala wysokości dźwięku mierzona metodą akustyki psychologicznej określająca subiektywny odbiór poziomu dźwięku przez ucho ludzkie względem obiektywnej skali pomiaru częstotliwości dźwięku w hercach. Na podstawie pomiarów przyjęto założenie, że ton o częstotliwości 1000 Hz przy poziomie ciśnienia akustycznego 40 dB powyżej progu słyszalności ma 1000 meli [2]. Zależność między skalą mel i Hz ma charakter nieliniowy i określa się ją wzorem [3]:

$$M(f) = 2595 \log_{10}(1 + f/700)\ [Mel] \tag{2.5}$$

Współczynniki mel cepstralne są szeroko wykorzystywane w dziedzinie akustyki. Niosą w sobie informacje odnośnie współczynników cepstralnych sygnału w dziedzinie melowej. Znajdują zastosowanie w przetwarzaniu sygnałów, mogą być wykorzystywane do parametryzacji wszelkich sygnałów akustycznych. Schemat obliczania współczynników MFCC przedstawiony jest na Rys. 2.2, a poszczególne jego etapy opisane są w dalszej części rozdziału.

Rys. 2.2. Schemat wyliczania współczynników MFCC

Sygnał dzielony jest na ramki odpowiadające 20 – 40 ms nagrania, dzięki czemu z sygnału szybkozmiennego powstaje quasi stacjonarny. Kolejne bloki dodawane są na zakładkę w taki sposób, że ramki czasowe na siebie nachodzą (Rys. 2.3) [5].



Rys. 2.3. Wizualizacja podziału sygnału na ramki na przykładzie nagrania interwału (dwóch występujących po sobie wysokości dźwięków)

Następnie sygnał zostaje okienkowany (np. oknem Hamminga). Z każdej ramki obliczana jest dyskretna transformata Fouriera, dzięki czemu uzyskana zostaje informacja odnośnie rozkładu amplitudowo-częstotliwościowego w analizowanym fragmencie nagrania [5]:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-i2\pi kn/N} \quad \text{dla } 1 \leq k \leq K \tag{2.6}$$

Na podstawie DFT wyliczane zostaje widmo mocy sygnału (Rys. 2.4) [5]:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \tag{2.7}$$

gdzie:

N – ilość próbek w jednej ramce sygnału,

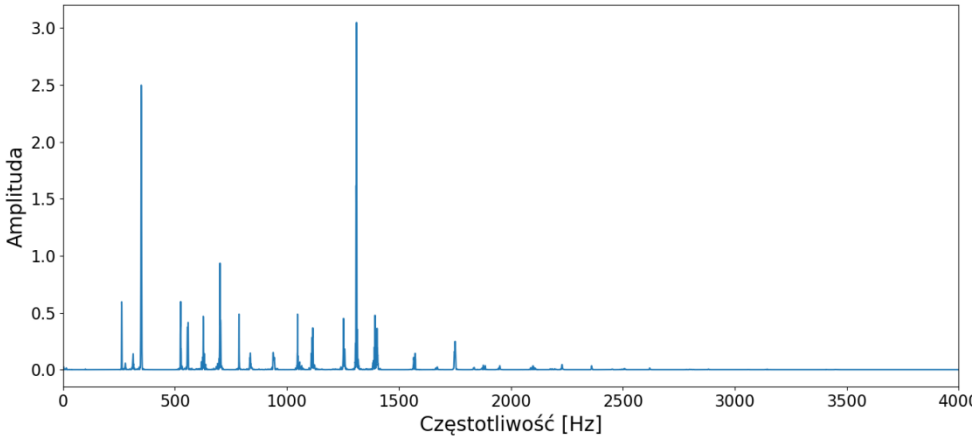
K – ilość próbek DFT,

$S_i(k)$ – wartość DFT dla i-tej ramki sygnału,

$s_i(n)$ – n-ta próbka i- tej ramki sygnału w dziedzinie czasowej,

$h(n)$ – okno czasowe (np. Hamming) o długości N,

$P_i(k)$ – widmo mocy sygnały dla i-tej ramki.



Rys. 2.4. Widmo mocy sygnału na przykładzie nagrania interwału z Rys. 2.3

Kolejnym krokiem jest przefiltrowanie widma mocy sygnału przez bank filtrów melowych. W efekcie uzyskuje się zsumowane wartości energii w kolejnych przedziałach wyznaczonych przez poszczególne filtry [5].

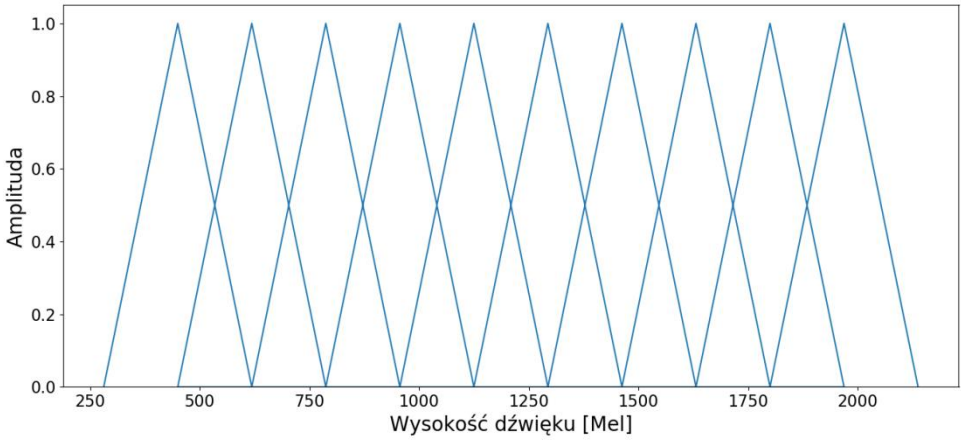
By wyznaczyć bank filtrów melowych należy wykonać następujące kroki [5]:

1. Wyznaczyć minimalną i maksymalną częstotliwość graniczną wszystkich filtrów.
2. Określić ilość filtrów.
3. Na podstawie poniższego wzorów wyznaczyć punkty graniczne (początek i koniec) dla wszystkich filtrów w dziedzinie meli:

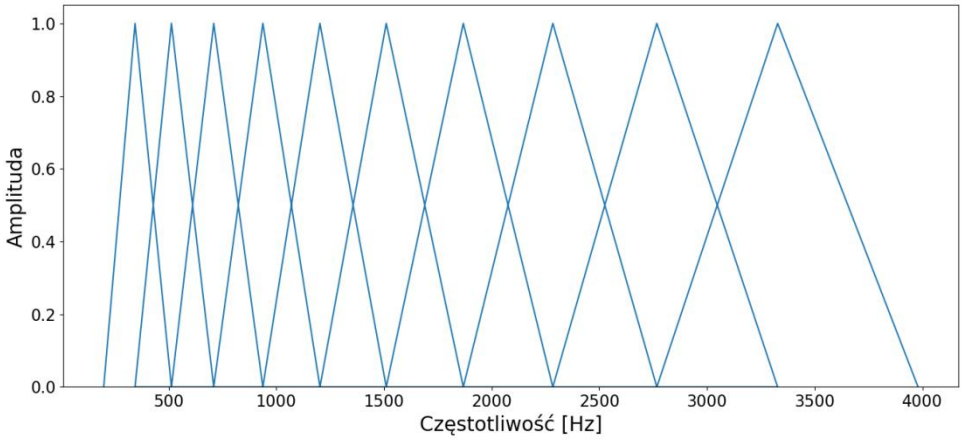
$$M(f) = 1125 \ln(1 + f/700) \ [Mel] \tag{2.8}$$

4. Przeliczyć wartości w melach na herce:

$$M(m) = 700\left(e^{m/1125} + 1\right) [Hz] \tag{2.9}$$



Rys. 2.5. Przykład banku filtrów w skali melowej



Rys. 2.6. Przykład tego samego banku filtrów w dziedzinie częstotliwości

Przyjmując równe przedziały banku filtrów w skali melowej, po przeliczeniu ich na skalę częstotliwości otrzyma się szersze filtry dla wyższych częstotliwości (Rys. 2.5, Rys. 2.6). W kontekście tej pracy ma to swoje zalety, gdyż pierwszych parę harmonicznych analizowanych sygnałów znajduje się w przedziale do około 1000Hz. Z uwagi na to, że harmoniczne niższych rzędów są lepiej widoczne na widmie sygnału (mają większą amplitudę), znajdują się w nich bardziej istotne informacje.

Otrzymana energia z przedziałów wyznaczonych przez bank filtrów melowych jest następnie logarytmowana. W ten sposób otrzymywane są współczynniki MFC (ang. *Mel Frequency Cepstrum*) [5].

Na koniec zostaje wyliczona dyskretna transformata cosinusowa. Głównym powodem tej operacji jest usunięcie części wspólnej stykających się filtrów melowych, tak by każdy filtr był niezależny względem innego. Wynikiem tej operacji są współczynniki MFCC [5].

Na podstawie wyliczonych współczynników MFCC można wyliczyć ich delty i delty-delt (wzór 2.10) [5]. Delty MFCC informują o zmienności sygnału. Przy obliczaniu delty uwzględniane są różnice wyników MFCC w dwóch sąsiednich ramkach czasowych. Delty-delt są powieleniem tej parametryzacji, ale zamiast różnic w MFCC brane są pod uwagę różnice w deltach.

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.10}$$

gdzie:

d_t – delta z współczynników MFCC w chwili czasowej t,

N – liczba współczynników MFCC (zazwyczaj 2),

c_t – współczynniki MFCC w chwili czasowej t.

2.3 GMM (ang. *Gaussian Mixture Model*)

Mikstura gaussowska jest to model próbujący znaleźć najlepszy dobór wielowymiarowych rozkładów prawdopodobieństwa określających pewien zbiór danych (wzór 2.11) [6,7].

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x}) \tag{2.11}$$

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}|\sum_i|^{1/2}} \exp\left\{-\frac{1}{2}(\vec{x} - \overrightarrow{\mu_i})' \sum_i^{-1} (\vec{x} - \overrightarrow{\mu_i})\right\} \tag{2.12}$$

gdzie:

$\vec{x}$ – D-wymiarowy wektor cech

M – liczba komponentów

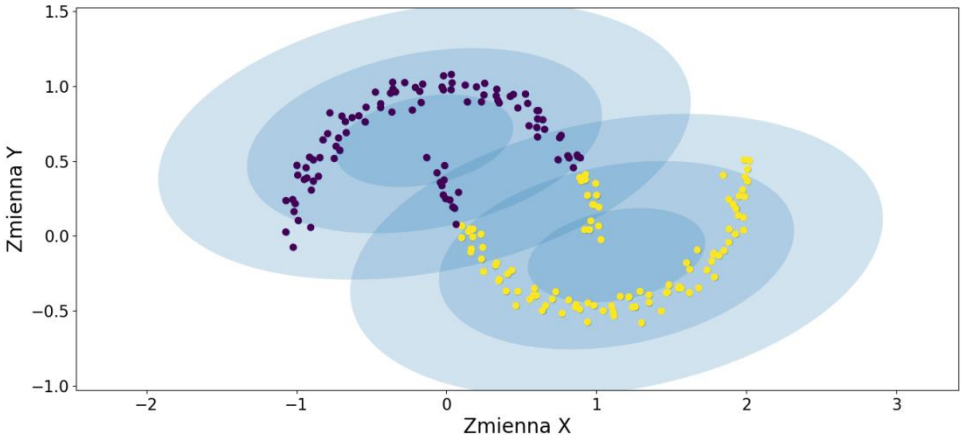
p_i – waga i-tego komponentu Gaussa ($\sum_{i=1}^{M} p_i = 1$)

$b_i(\vec{x})$ – i-ty komponent mikstury (zdefiniowany jako D-wymiarowa funkcja Gaussa)

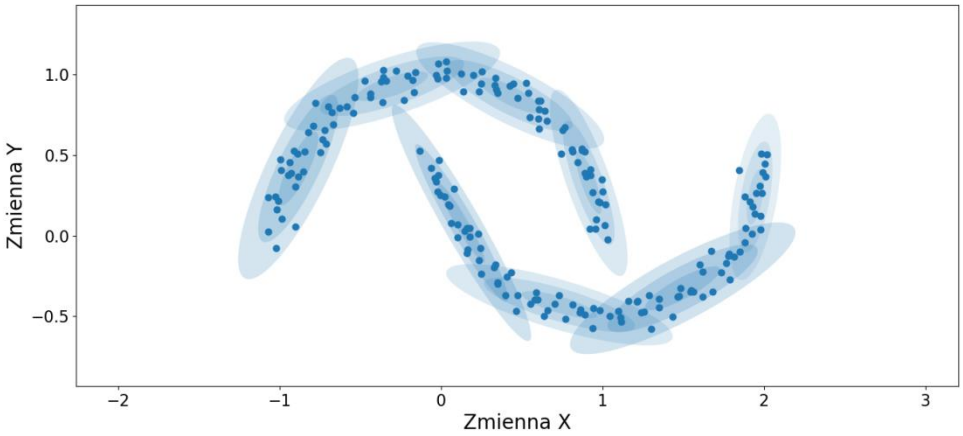
$\overrightarrow{\mu_i}$ – wektor wartości oczekiwanych rozkładu

$\sum_i$ – macierz kowariancji D-wymiarowej macierzy cech

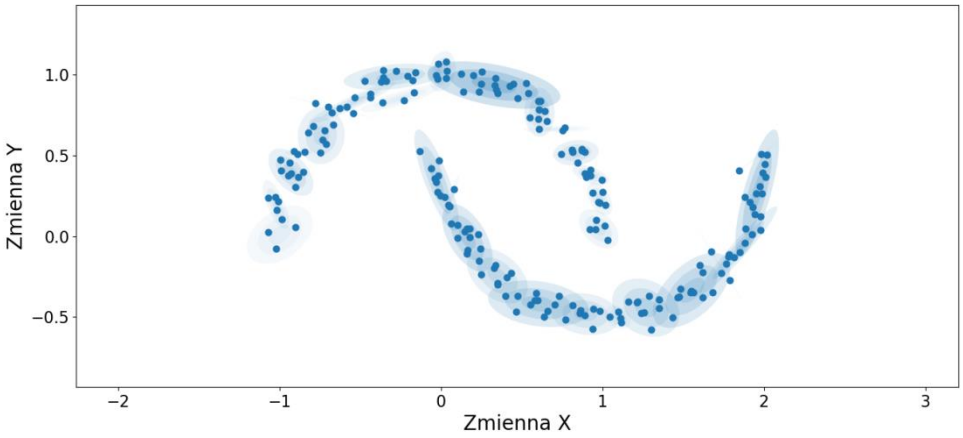
By model określony przez GMM poprawnie określał zbiór danych istotne jest wyznaczenie odpowiedniej liczby komponentów. Zbyt mała liczba spowoduje zbyt ogólne przybliżenie struktury (Rys. 2.7), za duża - nadmiarowe dopasowanie (Rys. 2.9). Sposobem na wyznaczenie liczby komponentów jest zastosowanie narzędzi analitycznych takich jak: AIC (ang. *Akaike information criterion*) oraz BIC (ang. *Bayesian information criterion*) (Rys. 2.10) [7].



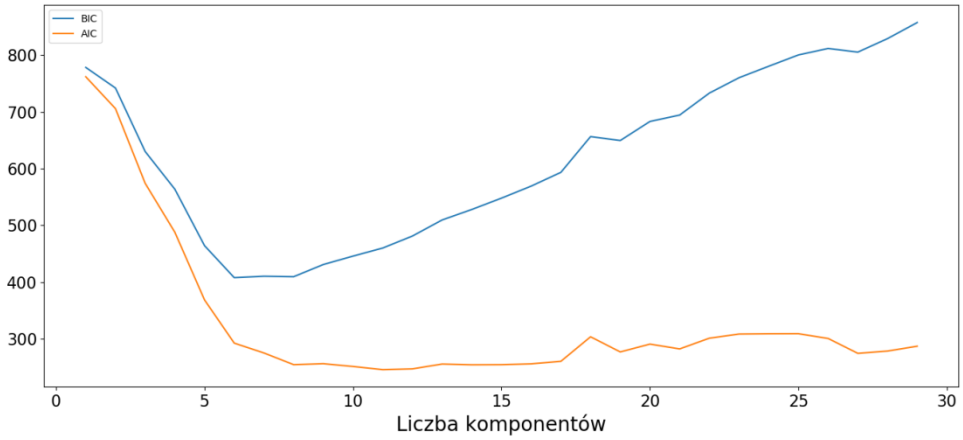
Rys. 2.7. Model GMM ze zbyt małą liczbą komponentów (2) względem zbioru danych [7]



Rys. 2.8. Model GMM z odpowiednią liczbą komponentów (8) względem zbioru danych [7]



Rys. 2.9. Model GMM ze zbyt dużą liczbą komponentów (30) względem zbioru danych [7]



Rys. 2.10. Współczynniki AIC i BIC dla ukazanego na Rys. 2.7 zbioru danych

Kryterium informacyjne Akaikego AIC to narzędzie analityczne służące do wyboru pomiędzy modelami statystycznymi o różnej liczbie predyktorów [8]. Im mniejsza wartość AIC tym model jest lepiej dopasowany.

$$AIC = -2\sum_{i} \ln(\hat{\pi}_i) + 2q \tag{2.13}$$

gdzie:

$\hat{\pi}_i$ - estymowane prawdopodobieństwo, przy założeniach danego modelu, uzyskania takiej właśnie wartości obserwacji i jaka była naprawdę uzyskana,

q – liczba parametrów modelu.

Bayesowskie kryterium informacyjne Schwartza BIC w modelowaniu równań strukturalnych jest to jeden ze wskaźników dopasowania modelu. Im mniejsza wartość BIC tym model jest lepiej dopasowany.

$$BIC = \ln(n)k - 2\ln(\hat{L}) \tag{2.14}$$

gdzie:

n – liczba obserwacji,

k – liczba parametrów modelu,

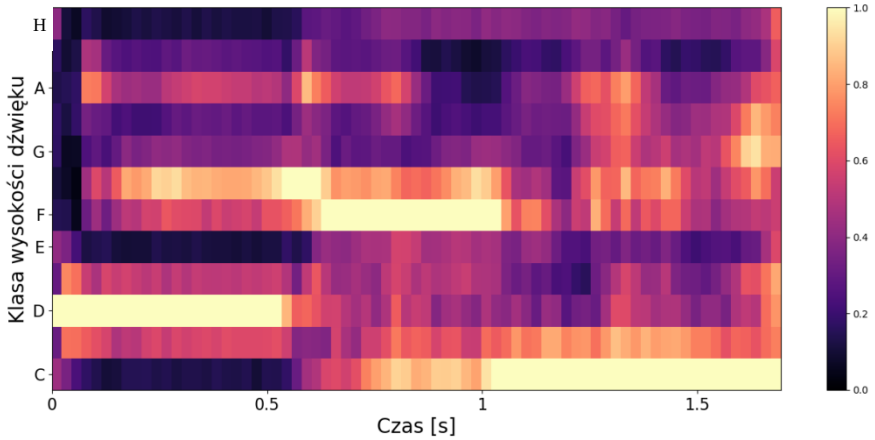
$\ln(\hat{L})$ – funkcja wiarygodności dla oszacowanego wektora parametrów.

3. Przykłady dostępnych rozwiązań

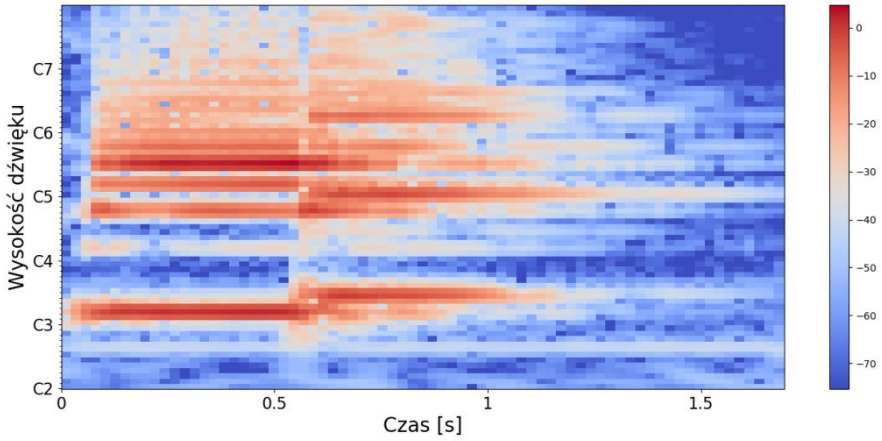
3.1 Librosa

Jest to biblioteka działająca w języku programowania Python skupiająca w sobie zagadnienia MIR. Nazwa librosa pochodzi od złączenia dwóch słów Lab + ROSA: the LABoratory for the Recognition and Organization of Speech and Audio at Columbia University, gdzie program był początkowo rozwijany [9]. Projekt skupia się na analizie sygnału audio w kontekście częstotliwości, dynamiki, rozkładu wielotonów harmonicznych, określenia tempa utworu, itp. Funkcjami, które odnoszą się bezpośrednio do tej pracy są:

- *librosa.feature.chroma_cqt* / *librosa.feature.chroma_stft* – funkcje, które analizują sygnał pod względem występujących w nim wysokości dźwięków. Określają, do jakich klas skali muzycznej przynależą kolejne ramki czasowe nagrania, nie determinując przy tym, w której oktawie dane wysokości się znajdują (Rys. 3.1),
- *librosa.core.cqt* – funkcja analizująca sygnał pod kątem prawdopodobieństwa wystąpienia wysokości dźwięku w krótkich chwilach czasowych (Rys. 3.2).



Rys. 3.1. Reprezentacja wyników uzyskanych przy zastosowaniu funkcji librosa.feature.chroma_cqt



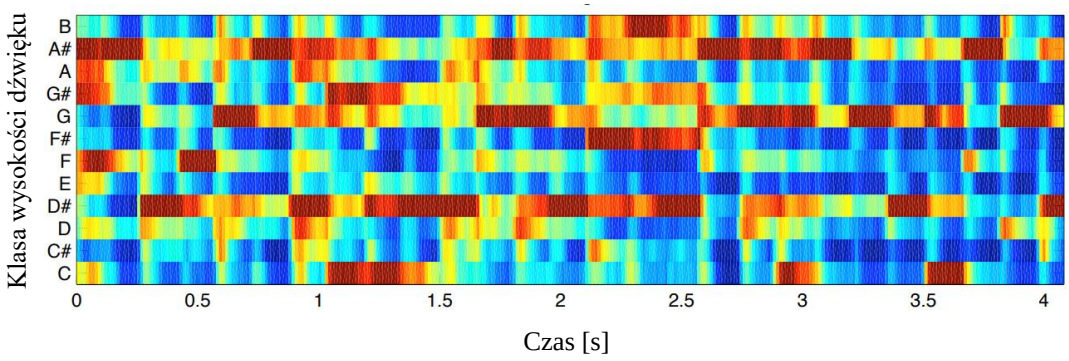
Rys. 3.2. Reprezentacja wyników uzyskanych przy zastosowaniu funkcji librosa.core.cqt

3.2 MIRtoolbox

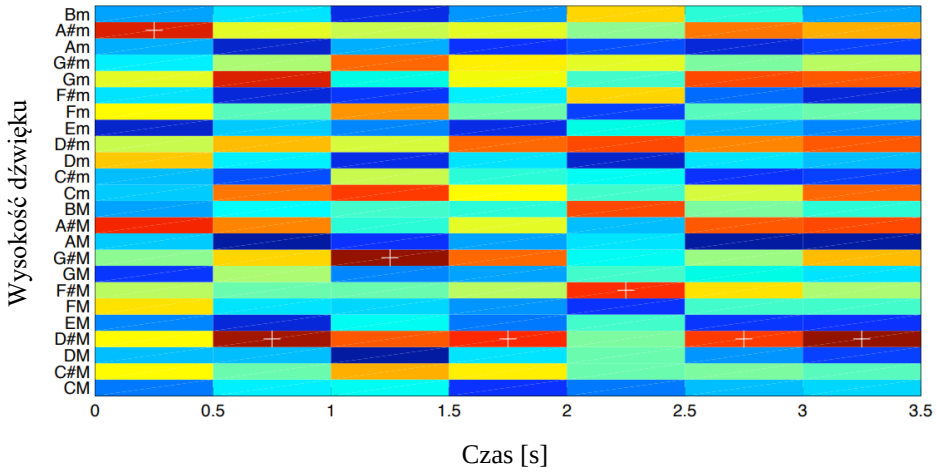
Jest to biblioteka działająca w języku programowania MATLAB skupiająca w sobie zagadnienia MIR. Zawiera funkcje poświęcone ekstrakcji informacji muzycznych z nagrania takich jak rytm, tonalność, rozkład harmonicznych itp.

Zastaw tych narzędzi został stworzony w ramach projektu "Brain Tuning" sfinansowanego przez Unię Europejską (FP6-NEST). Założeniami tego projektu było zbadanie relacji pomiędzy matematyczną analizą utworów muzycznych, a wzbudzanymi przez muzykę emocjami oraz ich wpływem na aktywność mózgu [10]. Funkcjami, które odnoszą się bezpośrednio do tej pracy są:

- *Mirchromagram* – funkcja, które przyporządkowuje próbki sygnału do klas wysokości dźwięków nie określając przy tym, w której oktawie dana wysokość się znajduje (Rys. 3.3).
- *Mirkey* – funkcja analizująca sygnał pod kątem prawdopodobieństwa występujących w nim tonalności (Rys. 3.4).



Rys. 3.3. Reprezentacja wyników uzyskanych przy zastosowaniu funkcji Mirchromagram [10]



Rys. 3.4. Reprezentacja wyników uzyskanych przy zastosowaniu funkcji Mirkey [10]

Przedstawione powyżej gotowe narzędzia, mimo ścisłego powiązania z tematem niniejszej pracy oraz szeroką gamę funkcji związanych z dziedziną MIR, bezpośrednio nie oferują gotowych algorytmów do segmentacji nagrań ze względu na pojawiające się w nich wysokości dźwięków. Przytoczone funkcje analizują sygnał audio pod kątem występujących w nim tonów, lecz nie określają konkretnie na jaką wysokość dźwięku wskazuje rozpoznanie w danej chwili czasowej nagrania.

Korzystając z licznych funkcji zawartych w bibliotece Librosa lub MIRtoolbox można napisać program działający w podobny sposób jak aplikacja stworzona na potrzeby tej pracy, lecz ze względu na zbyt uniwersalną funkcjonalność powyższych narzędzi, skuteczność ich działania w kontekście segmentacji byłaby niewystarczająca.

4. Budowa aplikacji

4.1 Plan działania algorytmów

Działanie aplikacji można podzielić na trzy etapy:

- utworzenie modeli wysokości na podstawie bazy danych,
- przygotowanie utworów pod analizę,
- dokonanie klasyfikacji i segmentacji.

Utworzenie modeli dźwięków

Dla każdego instrumentu (flet, obój, klarnet) została określona baza dźwięków pokrywająca cały zakres wysokości, jaki dany instrument może osiągnąć (Tab. 4.1).

Tab. 4.1. Zestawienie zarejestrowanych wysokości instrumentów

Instrument	Dostępna skala
Flet	c^1 - c^4
Klarnet	d - g^3
Obój	ais - g^3

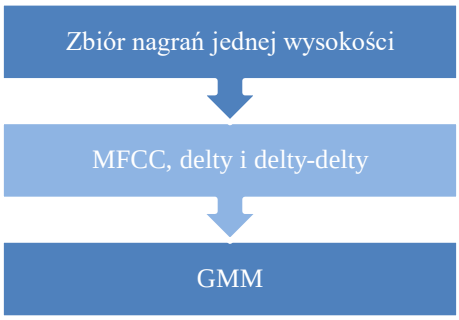
Próbki mają postać krótkich nagrań (około 2 s), gdzie każde z nich jest pojedynczym dźwiękiem odnoszącym się do jednej wysokości. Na jedną wysokość przypada
16 nagrań, które zostały stworzone w różnych warunkach za względu na: sposób mikrofonowania, dynamikę gry, sygnał z lewego bądź prawego kanału (podczas rejestracji próbek użyto dwóch mikrofonów by uzyskać obraz stereo). Ponadto każdą wersję umieszczoną w tabeli 4.2 nagrano dwukrotnie.

Tab. 4.2. Zestawienie wszystkich opcji rejestracji dźwięku dla pojedynczej wysokości

Mikrofonowanie	Bliskie				Dalekie			
Dynamika gry	Mezzo forte		Mezzo piano		Mezzo forte		Mezzo piano	
Kanał	Lewy	Prawy	Lewy	Prawy	Lewy	Prawy	Lewy	Prawy

By utworzyć wzór dźwięku dla danego instrumentu należy z wszystkich sygnałów związanych z daną wysokością wyliczyć współczynniki MFCC oraz delty i delty-delt. Następnie z danej macierzy powstaje baza danych, na podstawie której wylicza się model GMM.

Dany algorytm (Rys. 4.1) powtarza się dla całej skali dźwięków dostępnej dla danego instrumentu. Otrzymane wzorce posłużą przy dalszej klasyfikacji i segmentacji.

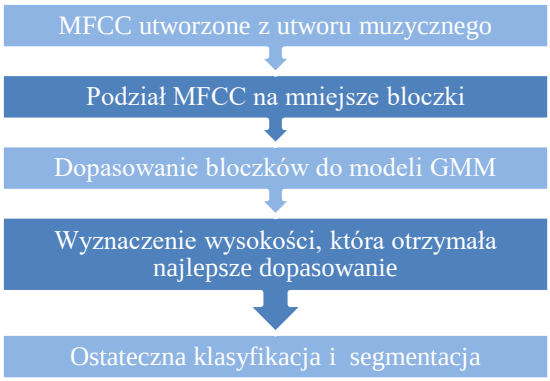


Rys. 4.1. Schemat tworzenia modelu jednej wysokości wybranego instrumentu

Przygotowanie utworów do analizy

By przygotować nagranie pod klasyfikację należy wyliczyć z niego współczynniki MFCC oraz delty tak, jak zostało to zrobione w poprzednim kroku.

Klasyfikacja i segmentacja utworów

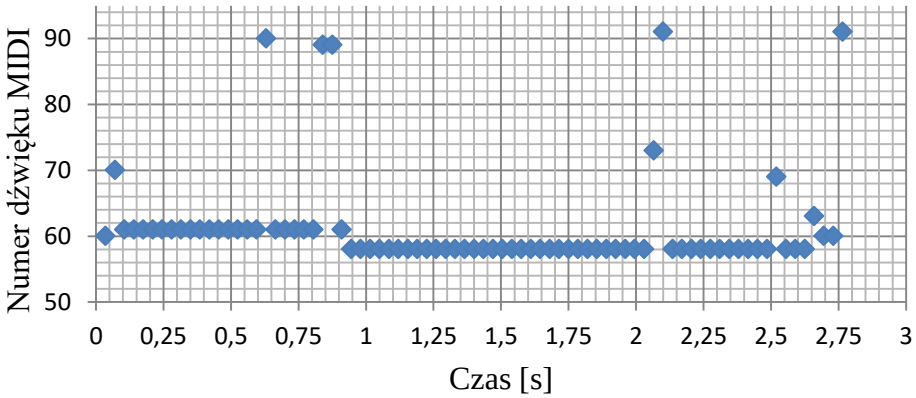


Rys. 4.2. Schemat klasyfikacji i segmentacji utworu po uprzednim stworzeniu modeli dźwięków oraz przygotowaniu nagrania do analizy

Pierwszym etapem klasyfikacji występujących w nagraniu wysokości dźwięków jest podzielenie uprzednio policzonych współczynników MFCC, delt, delty-delty na bloki składające się z próbek odpowiadających 35 ms sygnału. W następnym kroku porównuje się te fragmenty z modelami GMM wszystkich dostępnych wysokości. Porównanie, które otrzyma najlepsze dopasowanie jest zapisywane. Efektem tej operacji jest wektor rozpoznanych wysokości, którego każdy element odpowiada 35 ms nagrania (Rys. 4.3).

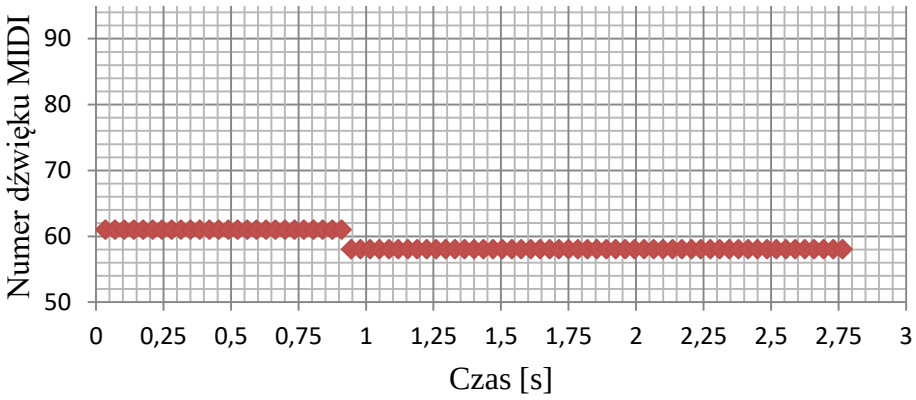
Istotną cechą przyjęcia takiego rozwiązania jest fakt, że im krótsze nagrania zostają poddane analizie (mniejsza ilość współczynników MFCC), tym algorytm jest mniej precyzyjny. Z drugiej strony zbyt duża liczba próbek sprawia, że rozdzielczość czasowa jest niewystarczająca, by automatycznie segmentować szybko zmienny sygnał. Zaprezentowane rozwiązanie jest kompromisem między poprawnością dopasowania, a rozdzielczością analizy.

Wynik pierwszej segmentacji zlicza się w grupy tych samych wysokości dźwięków występujących po sobie. Przyjęto założenie, że jeżeli liczność grupy jest mniejsza niż 5 to zostaje ona uznana za złe rozpoznanie i zakłada się, że powinna należeć do jednej z dwóch najbliższych poprawnie rozpoznanych wysokości. By wybrać, do której wyliczana jest różnica w półtonach między dobrze i źle rozpoznanymi tonami. Dźwięki zostają przypisane do tej grupy, która jest bliżej na skali muzycznej (Rys. 4.4). Tak dobrana liczność grupy odpowiada 175 ms nagrania, co dla gry w tempie 120 bpm[1] oznacza rozdzielczość około jednej szesnastki.



Rys. 4.3. Efekt pierwszego dopasowania wysokości dźwięków w nagraniu

[1] Bpm (ang. *beats per minute*) - miara tempa utworu muzycznego, wyznaczająca liczbę ćwierćnut przypadających na jedną minutę sygnału.



Rys. 4.4. Efekt ostatecznego dopasowania wysokości dźwięków w nagraniu

4.2 Konfiguracja środowiska

Aplikacja stworzona na potrzeby niniejszej pracy została napisana w języku programowania Python. Wybór tego języka jest uzasadniony jego rosnącą popularnością oraz mnogością dostępnych narzędzi, szczególnie w kontekście maszynowego uczenia. Jako środowisko programistyczne użyto JetBrains PyCharm wraz z doinstalowaną dystrybucją Anaconda, w której skład wchodzi większość najczęściej wykorzystywanych bibliotek. W tej pracy wykorzystane zostały:

- Numpy – obsługująca operacje na tablicach wielu typu zmiennych,
- Matplotlib – służąca do tworzenia wykresów,
- Scikit – zawierająca narzędzia wykorzystywane w maszynowym uczeniu.

4.3 Dodatkowe biblioteki

Soundfile - biblioteka służy do czytania i zapisywania plików audio. Python oferuje wiele narzędzi, które mają taką samą funkcjonalność, lecz ta w porównaniu do wielu, obsługuje pliki w formacie 24 bitów, co jest istotne w kontekście tej pracy.

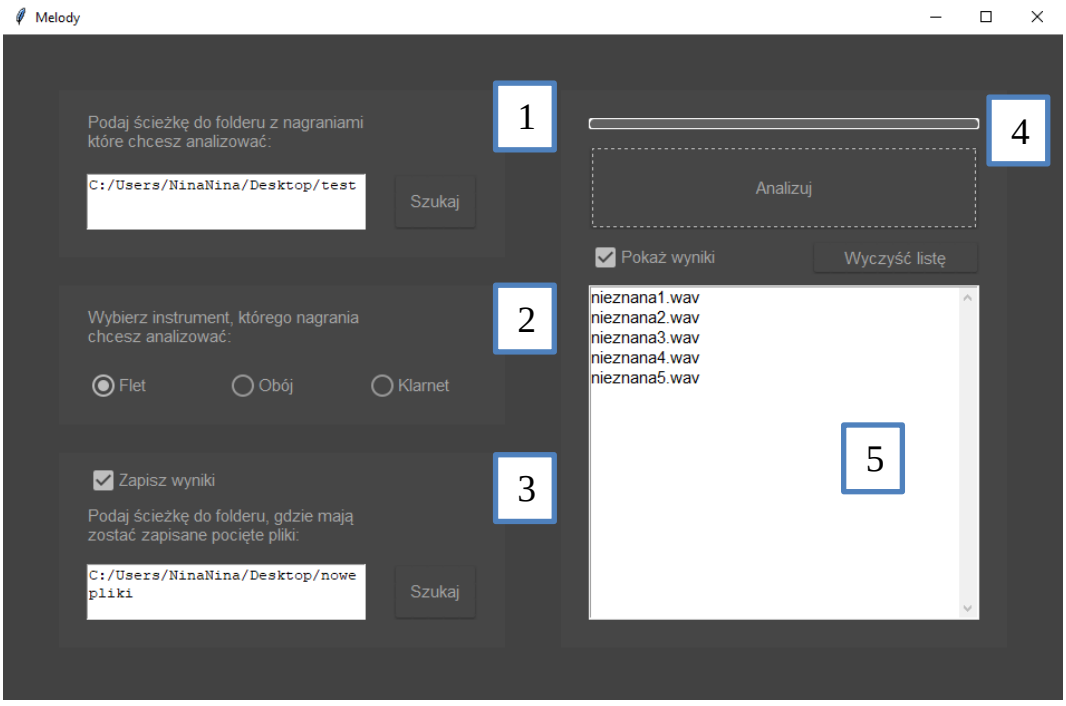
Tkinter - jest to jedna z najbardziej podstawowych bibliotek do obsługi interfejsu aplikacji w języku Python. Skorzystano z niej w tej pracy, gdyż cechuje się prostotą w obsłudze, posiada obszerną dokumentację oraz wiele gotowych przykładów zastosowań.

Ttkthemes - rozszerzenie biblioteki tkinter o możliwość korzystania z różnych szablonów graficznych aplikacji.

Python speech features – biblioteka służąca do parametryzacji sygnału audio, zawiera w sobie funkcje wyliczające współczynniki MFCC itp.

4.4 Interfejs graficzny aplikacji

W celu wygodnego posługiwania się stworzoną w ramach tej pracy aplikacją zaprojektowano jej interfejs graficzny. Zwrócono uwagę, by był on jak najbardziej intuicyjny i posiadał minimalną ilość funkcji tak, aby użytkownik mógł w prosty sposób obsłużyć program. W dalszej części rozdziału przybliżono poszczególne składowe interfejsu ukazanego na rysunkach 4.5 – 4.6.



Rys. 4.5. Interfejs aplikacji

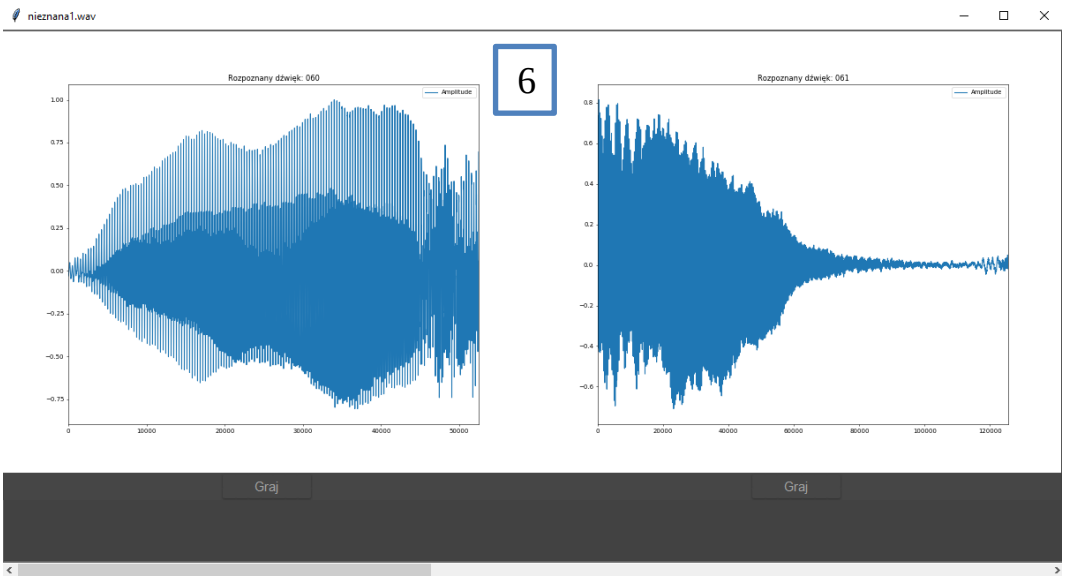
1. Obszar wyboru folderu, w którym znajdują się nagrania utworów, które użytkownik chce analizować.

2. Obszar wyboru instrumentu, którego nagrania mają zostać automatycznie posegmentowane ze względu na wysokość dźwięku.

3. Obszar wyboru folderu, w którym mają zostać zapisane posegmentowane nagrania. Pole wyboru „Zapisz wyniki” automatycznie po uruchomieniu aplikacji jest odznaczone, przez co pole wyboru ścieżki do zapisania plików jest nieaktywne. Jeśli użytkownik nie zaznaczy pola „Zapisz wyniki”, po skończeniu analizy ścieżek nie zostaną one zapisane na dysku.

4. Po wybraniu ścieżek oraz instrumentu użytkownik powinien kliknąć przycisk „Analizuj”, co rozpocznie automatyczną segmentację nagrań ze względu na wysokość dźwięku. W zależności od ilości i długości nagrań poddanych do analizy proces ten może być czasochłonny. Nad przyciskiem znajduje się pasek postępu informujący o postępie prac.

5. Pole wyboru „Pokaż wyniki” automatycznie po uruchomieniu aplikacji jest zaznaczone, przez co lista, w której znajdą się wyniki segmentacji jest aktywna i po skończeniu analizy próbek zostanie wypełniona przeanalizowanymi ścieżkami. Jeśli użytkownik nie chcę zobaczyć wyników segmentacji powinien odznaczyć pole „Pokaż wyniki”. Po dwukrotnym kliknięciu na nazwę pliku znajdującego się na liście, otworzy się nowe okno reprezentujące wyniki segmentacji (Rys. 4.6).



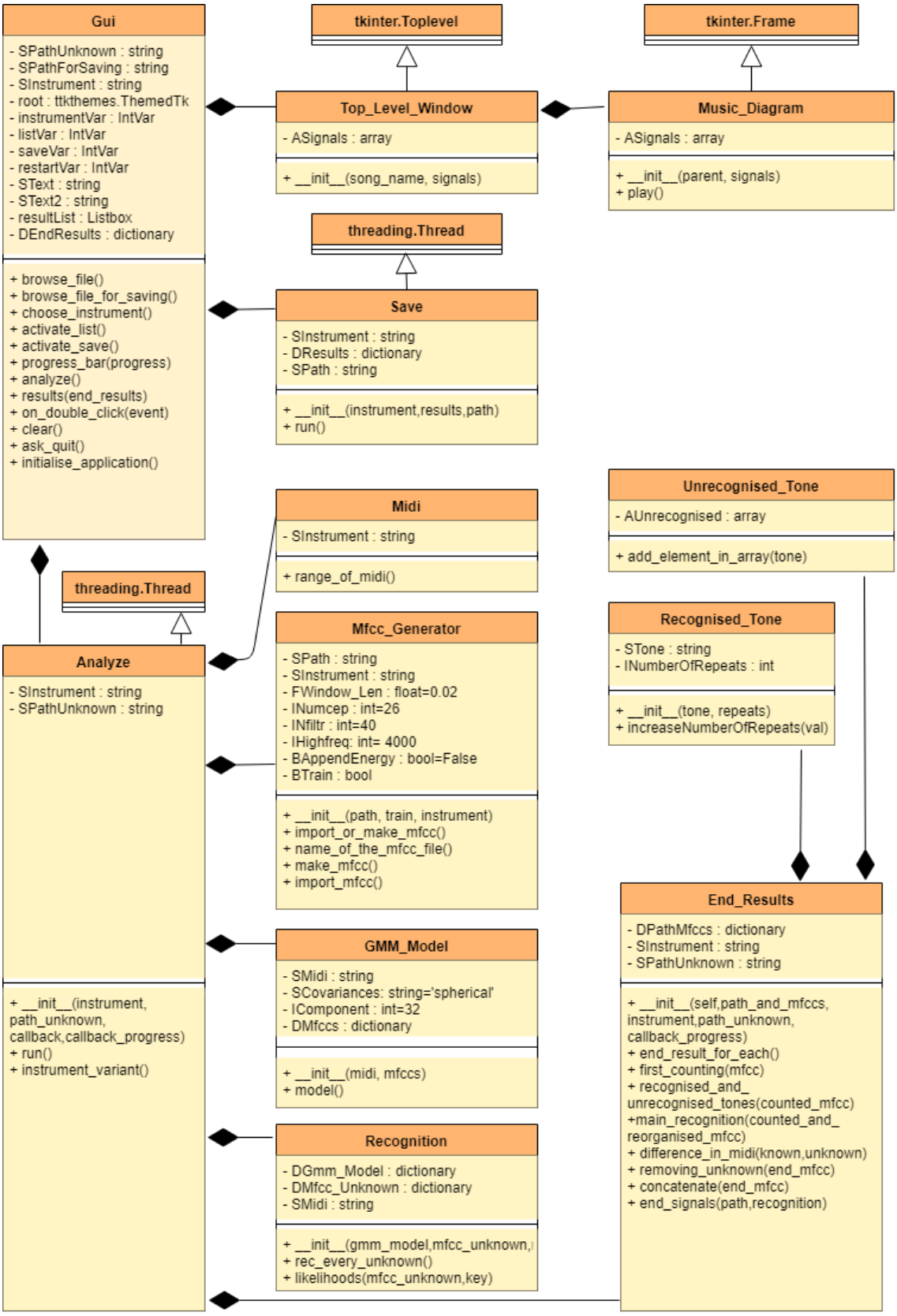
Rys. 4.6. Okno prezentacji wyników segmentacji jednego nagrania

6. Obszar prezentacji wyników segmentacji. Pasek tytułu odnosi się do nazwy analizowanego pliku. Tutułami wykresów są ropoznane dźwięki w formacie MIDI[2]. Pod wykresami znajdują się przyciski „Graj”, gdy użytkownik naciśnie któryś z nich usłyszy jak brzmi fragment nagrania zaprezentowany na wykresie nad nim.

4.5 Struktura danych

Aplikacja składa się z dwunastu klas, których relacje zaprezentowane są na rysunku 4.7. W kolejnych rozdziałach skupiono się na opisaniu kluczowych funkcji odpowiedzialnych za działanie programu.

[2] MIDI (ang. *Musical Instrument Digital Interface*) – format zapisu wysokości dźwięków w systemie równomiernie temperowanym (o stroju a^1=440Hz) jako numery, przykładowo: gis^1=68, a^1=69, ais^1=70.



Rys. 4.7. Diagram klas aplikacji

4.5.1 Klasa Gui, Top_Level_Window, Music_Diagram

Wszystkie z tych klas odpowiadają za interfejs graficzny aplikacji. Klasa Gui implementuje główne okno aplikacji oraz jest łącznikiem między działaniami użytkownika, a wykonywaniem się części kodu odpowiedzialnej za segmentację utworów, zapisywanie oraz pokazywanie wyników analizy (Tab. 4.3).

Top_Level_Window pozwala na utworzenie nowego okna aplikacji. Obiekt tej klasy powstaje w momencie, gdy użytkownik dwukrotnie kliknie na nazwę utworu z listy wyników (Tab. 4.4).

W nowym oknie powstaje obiekt klasy Music_Diagram, który jest odpowiedzialny za demonstrację wyników segmentacji w formie wykresu oraz obsługuje funkcję, która pozwala na odsłuchanie każdego fragmentu pociętego nagrania (Tab. 4.5).

Tab. 4.3. Funkcje wchodzące w skład klasy Gui

Nazwa funkcji (parametry wejściowe)	**Opis działania funkcji**
browse_file()	Zapisuje ścieżkę dostępu do folderu z plikami, które użytkownik chce poddać analizie.
browse_file_for_saving()	Zapisuje ścieżkę dostępu do folderu, w którym mają zostać zapisane fragmenty pociętych plików.
choose_instrument()	Na podstawie wybranego przez użytkownika instrumentu, którego nagrania chce analizować, zapisuje nazwę instrumentu.
activate_list()	Jeśli przycisk „Pokaż wyniki” jest zaznaczony, to aktywuje okienko z listą w celu późniejszego wypełnienia jej wynikami. Jeśli nie, to lista nie jest aktywna.
activate_save()	Jeśli przycisk „Zapisz wyniki” ” jest zaznaczony, to aktywuje możliwość wyboru folderu, w którym mają zostać zapisane pliki.
progress_bar(progress)	Jako parametr otrzymuje procent postępu analizy plików, służy do obsługi paska postępu.
analyze()	Po przyciśnięciu przycisku „Analizuj” w osobnym wątku tworzy obiekt klasy Analyze. Przed utworzeniem obiektu klasy sprawdza, czy użytkownik wybrał instrument i ścieżkę do plików.

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
results(end_results)	Jako parametr przyjmuje wyniki analizy, które uzyskano z operacji na obiekcie klasy Analyze. Umieszcza każdy z przeanalizowanych utworów w oknie listy udostępniając w ten sposób możliwość zobaczenia wyników w nowym oknie programu. Jeżeli użytkownik zaznaczył, że chce zapisać wyniki analizy tworzy obiekt klasy Save w osobnym wątku.
on_double_click(event)	Obsługuje funkcjonalność dwukrotnego kliknięcia w wybrany plik z listy, po czym tworzy obiekt klasy Top_Level_Window, a w jego konstruktorze przekazuje nazwę wybranego pliku oraz wyniki segmentacji danego nagrania.
clear()	Po przyciśnięciu przycisku “Wyczyść listę” usuwa wszystkie pozycje z listy.
ask_quit()	Obsługuje poprawne zamykanie się programu. Po kliknięciu przycisku “Zamknij”, sprawdza czy wszystkie procesy zostały zakończone, a niepotrzebne pliki usunięte.
initialise_application()	Jest to funkcja odpowiedzialna za tworzenie wszystkich przycisków, list, pól tekstowych itp. okna aplikacji. Przechowuje informacje o ich położeniu, rozmiarze i funkcjonalności.

Tab. 4.4. Funkcje wchodzące w skład klasy Top_Level_Window

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(song_name, signals)	Konstruktor klasy, tworzy nowe okno programu, którego zawartością paska tytułu jest nazwa pliku. Dla każdej rozpoznanej wysokości w utworze tworzy obiekt klasy Music_Diagram a w jego konstruktorze przekazuje wyniki segmentacji danego nagrania odpowiadające jednemu rozpoznanemu dźwiękowi.

Tab. 4.5. Funkcje wchodzące w skład klasy Music_Diagram

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(parent, signals)	Konstruktor klasy, tworzy wykres z sygnału odpowiadającemu rozpoznanej wysokości oraz przycisk „Graj”, który umożliwia odsłuchanie ukazanego fragmentu.
play()	Gdy użytkownik kliknie przycisk „Graj” odtwarza fragment nagrania.

4.5.2 Klasa Analyze

Jest to klasa odpowiedzialna za przebieg analizy pod kątem segmentacji utworów. Jej głównym zadaniem jest tworzenie obiektów klas, które zwracają do niej wartości niezbędne do tworzenia obiektów kolejnych klas, by w finalnym działaniu zwrócić posegmentowane nagrania do klasy Gui w celu wyświetlenia lub zapisania wyników (Tab. 4.6).

Tab.4.6. Funkcje wchodzące w skład klasy Analyze

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(instrument, path_unknown, callback, callback_progress)	Konstruktor klasy, przy tworzeniu przekazywane są wartości wybranego instrumentu, ścieżki do analizowanych plików, funkcje wywołania zwrotnego wyników analizy oraz aktualizowania paska postępu.
run()	Kolejno tworzy obiekty klas: - Mfcc_Generator, - Midi, - GMM_Model, - Recognition, - End_Results. Zwraca do klasy Gui posegmentowane nagrania ze względu na występujące w nich wysokości.
instrument_variant()	Tworzy obiekty klasy Midi oraz Mfcc_Generator dla analizowanych plików i plików testowych.

4.5.3 Klasa Midi

Klasa pomocnicza przechowująca informacje odnośnie dostępnej skali dźwięków dla każdego instrumentu. Po jej utworzeniu zwraca do klasy Analyze, zakres dźwięków (Tab. 4.7).

Tab. 4.7. Funkcje wchodzące w skład klasy Midi

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(instrument)	Konstruktor obiektu klasy, przy tworzeniu przekazywana jest nazwa wybranego instrumentu.
range_of_midi()	Na podstawie wybranego przez użytkownika instrumentu, zwraca dostępny dla niego zakres dźwięków na skali muzycznej.

4.5.4 Klasa Mfcc_Generator

Klasa, która tworzy współczynniki MFCC, delty i delty-delt i zapisuje je do pliku lub pobiera z wcześniej obliczonego pliku. Po jej utworzeniu zwraca do klasy Analyze współczynniki MFCC, delty i delty-delt dla zbioru nagrań testowych i analizowanych (Tab. 4.8).

Tab. 4.8. Funkcje wchodzące w skład klasy Mfcc_Generator

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(path,train, instrument)	Konstruktor klasy, przy tworzeniu przekazywane są: ścieżka do plików, zmienna określająca czy jest to ścieżka do sygnałów testowych czy analizowanych, wybrany instrument.
import_or_make_mfcc()	Sprawdza czy z podanej ścieżki dostępu zostały już policzone pliki MFCC, jeśli tak to wykonuję funkcję import_mfcc(). Jeśli nie, funkcję make_mfcc().
name_of_the_mfcc_file()	W zależności od tego czy ścieżka odnosi się do plików testowych czy analizowanych zwraca nazwę pliku MFCC.
import_mfcc()	Importuje obliczone wcześniej współczynniki.

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
make_mfcc()	Z wszystkich plików typu wav w podanej ścieżce dostępu tworzy współczynniki MFCC, delty i delty-delt . Zapisuje dane na dysku i wywołuje funkcję import_mfcc().

4.5.5 Klasa GMM_Model

Klasa, która wylicza modele GMM wszystkich dostępnych dźwięków dla danego instrumentu, a następnie zwraca je do klasy Analyze (Tab. 4.9).

Tab. 4.9. Funkcje wchodzące w skład klasy GMM_Model

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(midi, mfccs)	Konstruktor klasy, przy tworzeniu przekazywane są: dostępna skala wysokości dźwięków dla wybranego instrumentu oraz pliki MFCC, delty i delty-delt utworzone z bazy plików testowych.
model()	Dla każdej dostępnej wysokości dźwięku obliczany jest model GMM. Z wszystkich modeli tworzona jest tablica, która przekazywana jest z powrotem do klasy Analyze.

4.5.6 Klasa Recognition

Na podstawie dostępnej skali dźwięków dla wybranego instrumentu, wyliczonych współczynników MFCC, delt i delty-delt utworów, które mają zostać poddane segmentacji oraz wszystkich modeli wysokości GMM, klasa Recognition dokonuje pierwszej analizy wysokości dźwięków w nagraniach. Wynikiem tej segmentacji jest wektor propozycji rozpoznanych dźwięków, gdzie każda próbka odpowiada 35 ms nagrania (Tab. 4.10).

Tab. 4.10. Funkcje wchodzące w skład klasy Recognition

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(gmm_model, mfcc_unknown, midi)	Konstruktor klasy, przy tworzeniu przekazywane są: modele wysokości GMM, pliki MFCC, delty i delty-delt utworzone z bazy plików analizowanych oraz dostępna skala wysokości dźwięków dla wybranego instrumentu.
rec_every_unknown()	Dla każdego nagrania, które zostało wybrane do przeanalizowania wykonuje funkcję likelihoods().
likelihoods(mfcc_unknown, key)	Dzieli współczynniki MFCC, na bloki, które odpowiadają długości nagrania wynoszącej 35 ms. Następnie sprawdza dopasowanie tych współczynników do modeli dźwięków GMM. Wysokość, która otrzyma największe prawdopodobieństwo (najlepsze dopasowanie do modelu) jest uznawana za tą, która występuję w tej chwili czasowej nagrania. Każde nagranie w efekcie działania tej funkcji otrzymuje wektor rozpoznanych wysokości.

4.5.7 Klasa End_Results, Recognised_Tone i Unrecognised_Tone

Na podstawie uzyskanych z Klasy Recognition propozycji jakie dźwięki są najbardziej prawdopodobne w krótkich chwilach czasowych (35 ms), Klasa End_Results grupuje większe skupiska dźwięków w jedną rozpoznaną wysokość. Fragmenty nagrania, które zdefiniowane zostały jako źle rozpoznane przypisuje do jednej z dwóch najbliższych grup. W rezultacie segmentuje nagranie na fragmenty z rozpoznaną wysokością (tab.4.11). Klasy Recognised_Tone i Unrecognised_Tone są pomocniczymi klasami, dzięki którym łatwiej przebiega segmentacja.

Tab. 4.11. Funkcje wchodzące w skład klasy End_Results

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(path_and_mfccs, instrument, path_unknown, callback_progress)	Konstruktor klasy, przy tworzeniu przekazywane są: wektory rozpoznanych wysokości we wszystkich nagraniach, wybrany instrument, ścieżka do folderu, w którym są analizowane pliki.

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
first_counting(mfcc)	Funkcja, która na podstawie wektora wstępnie rozpoznanych wysokości grupuje te same dźwięki występujące po sobie określając ich liczność.
end_result_for_each()	Metoda, która dla każdego nagrania wywołuje funkcje: - first_counting(), - recognised_and_unrecognised_tones(), - main_recognition(), - removing_unknown(), - concatenate(), - end_signals(). Zwraca do klasy Analyze wyniki analizy.
recognised_and_ unrecognised_tones (counted_mfcc)	Na podstawie pogrupowanych wysokości, korzystając z założenia, że grupy mniej liczne niż 5 są nieprawidłowo rozpoznane, tworzy obiekty klas Recognised_Tone i Unrecognised_Tone. Zwraca wektor obiektów tych klas.
main_recognition (counted_and_ reorganised_mfcc)	Przyporządkowuje dźwięki określone jako obiekty klasy Unrecognised_Tone do jednej z dwóch najbliższych poprawnie rozpoznanych grup. By wybrać do której wywołuje funkcję difference_in_midi(). Dźwięki zostają przypisane do tej wysokości, która jest bliżej na skali muzycznej.
difference_in_midi(known, unknown)	Oblicza różnicę w półtonach między dźwiękami rozpoznanymi poprawnie i niepoprawnie.
removing_unknown (end_mfcc)	Usuwa niepoprawnie rozpoznane wysokości dźwięków z wektora.
concatenate(end_mfcc)	Jeżeli sąsiadujące ze sobą rozpoznane wysokości są tą samą wysokością, to są łączy je w jedność.
end_signals(path, recognition)	Zwraca do funkcji end_result_for_each() pocięte na fragmenty nagranie wraz z określeniem rozpoznanej wysokości.

4.5.8 Klasa Save

Klasa, która wykonuję się tylko wtedy, gdy użytkownik zaznaczył opcję, że chcę zapisać na dysku wyniki analizy. Służy do zapisywania posegmentowanych nagrań (Tab. 4.12).

Tab. 4.12. Funkcje wchodzące w skład klasy Save

Nazwa funkcji (parametry wejściowe)	Opis działania funkcji
__init__(instrument, results, path)	Konstruktor klasy, przy tworzeniu przekazywane są: wybrany instrument, wyniki segmentacji, ścieżka do folderu, w którym mają zostać zapisane pliki.
run()	Każda rozpoznana wysokość, we wszystkich nagraniach, które zostały poddane analizie zapisywana jest do podanego w konstruktorze folderu. Nazwa zapisywanego pliku tworzona jest według reguły: Nazwa instrumentu_rozpoznana wysokość w formacie midi_nazwa pliku, który został poddany analizie_numer kolejności rozpoznanej wysokości.wav (np. flet_060_nieznana1_part1.wav)

5. Wyniki testów

By określić skuteczność aplikacji przeprowadzono testy na nagraniach, których rozkład wysokości dźwięków jest znany. Jako baza posłużyły nagrania interwałów, składające się z dwóch wysokości dźwięków, różniących się określoną liczbą półtonów, granych zaraz po sobie. Wśród nich połowa próbek grana była w umiarkowanym tempie wynoszącym 60 bpm, druga część dwukrotnie szybciej w tempie 120 bpm.

Dodatkowo sprawdzono skuteczność na nagraniach tetrachordów, które składały się z pięciu wysokości granych zaraz po sobie. W zależności od instrumentu dysponowano różną ilością nagrań ze względu na inne zakresy skali dźwięków możliwych do zagrania. By ocenić poprawność rozpoznania przyjęto założenia, że za każdą w pełni dobrze rozpoznaną wysokości przyznano punkt, następnie określono procent poprawnych odpowiedzi na podstawie wszystkich możliwych do zdobycia punktów. Wszelkie wyniki różniące się nawet jednym półtonem od prawidłowego rozpoznania zostały uznane za niepoprawne.

Ostatnim przeprowadzonym testem było porównanie wyników działania aplikacji z analizą wysokości dźwięków w nagraniach za pomocą funkcji z biblioteki Librosa (Rys. 5.1 – 5.3).

Tab. 5.1. Wyniki testów skuteczności aplikacji dla klarnetu na podstawie nagrań interwałów i tetrachordów

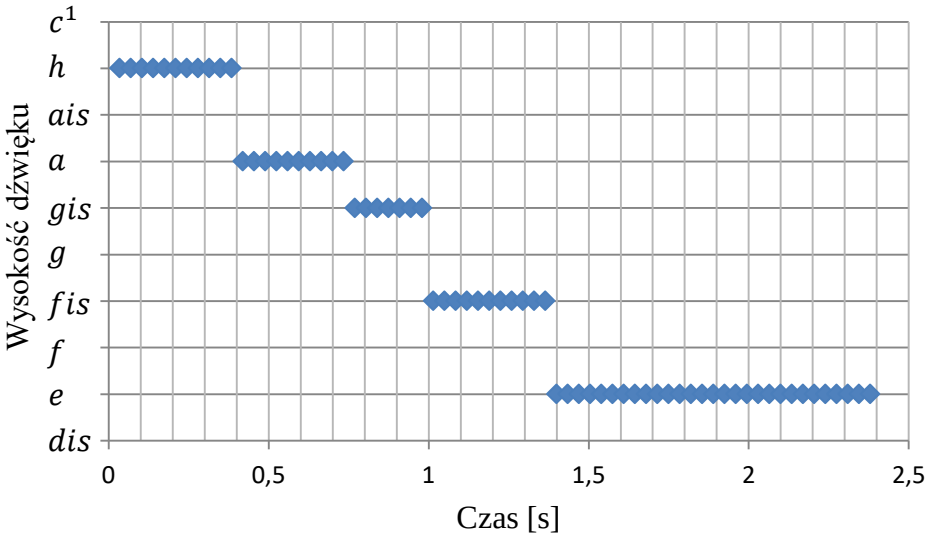
Analizowane nagrania	Skuteczność wolnych nagrań	Skuteczność szybkich nagrań	Całkowita skuteczność	Całkowita liczba nagrań
Sekunda mała	93%	94%	93%	2624
Sekunda wielka	87%	88%	87%	2560
Tercja mała	88%	88%	88%	1248
Tercja wielka	87%	88%	87%	1216
Tetrachord	-	-	61%	560

Tab. 5.2. Wyniki testów skuteczności aplikacji dla oboju na podstawie nagrań interwałów i tetrachordów

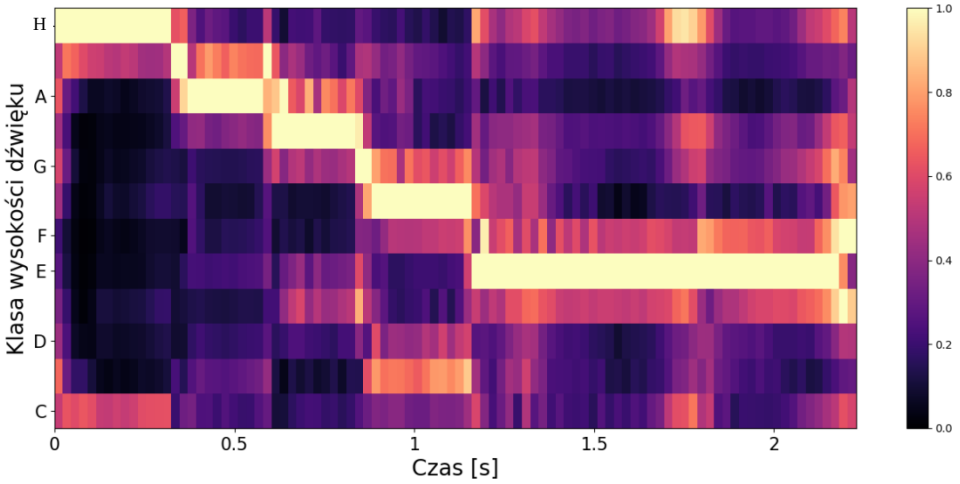
Analizowane nagrania	Skuteczność wolnych nagrań	Skuteczność szybkich nagrań	Całkowita skuteczność	Całkowita liczba nagrań
Sekunda wielka	87%	75%	81%	2048
Tercja mała	87%	74%	81%	992
Tercja wielka	93%	70%	81%	960
Kwarta czysta	88%	75%	81%	928
Tryton	93%	75%	84%	896
Kwinta czysta	89%	75%	82%	864
Seksta mała	93%	78%	85%	832
Seksta wielka	91%	76%	84%	800
Septyma mała	94%	77%	85%	768
Septyma wielka	93%	74%	84%	736
Oktawa czysta	94%	79%	87%	704
Tetrachord	-	-	64%	432

Tab. 5.3. Wyniki testów skuteczności aplikacji dla fletu na podstawie nagrań interwałów i tetrachordów

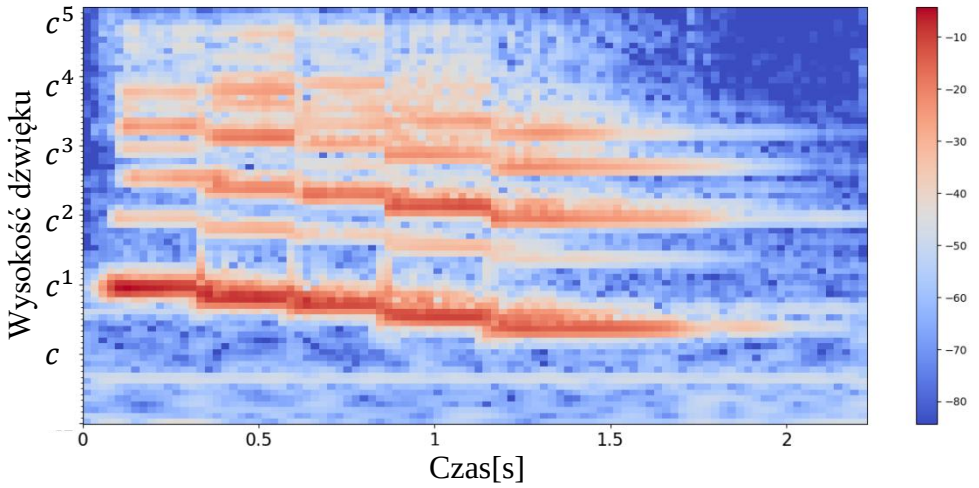
Analizowane nagrania	Skuteczność wolnych nagrań	Skuteczność szybkich nagrań	Całkowita skuteczność	Całkowita liczba nagrań
Sekunda mała	87%	82%	85%	2304
Sekunda wielka	79%	67%	73%	2240
Tercja mała	82%	74%	78%	1088
Tercja wielka	65%	72%	68%	1056
Kwarta czysta	84%	79%	81%	1024
Tryton	84%	77%	81%	992
Kwinta czysta	82%	79%	80%	960
Seksta mała	81%	68%	75%	928
Seksta wielka	82%	70%	76%	896
Septyma mała	79%	77%	78%	864
Septyma wielka	80%	78%	79%	832
Oktawa czysta	80%	81%	80%	800
Tetrachord	-	-	62%	476



Rys. 5.1. Demonstracja działania aplikacji na przykładzie tetrachordu



Rys. 5.2. Demonstracja działania funkcji librosa.feature.chroma_cqt na przykładzie tetrachordu



Rys. 5.3. Demonstracja działania funkcji librosa.core.cqt na przykładzie tetrachordu

6. Wnioski końcowe

Celem niniejszej pracy było stworzenie aplikacji służącej do automatycznej segmentacji wielonutowych próbek instrumentów dętych ze względu na występujące w nich wysokości dźwięków. Jednym z głównych założeń było określenie rozwiązań, dzięki którym w prosty sposób można by powiększać bibliotekę analizowanych instrumentów, bez konieczności przebudowywania całej aplikacji. Uzyskano ten efekt poprzez zastosowanie metod maszynowego uczenia. Na podstawie wyników zamieszczonych w tabelach 5.1 – 5.3 widać, że w przypadku testu na interwałach algorytm uzyskał większą skuteczność dla nagrań, w których tempo zmiany dźwięków było wolniejsze. Poprawne rozpoznanie tetrachordów w okolicach 60% jest spowodowane ich szybszym tempem zmian niż 60 bpm oraz specyfiką przeprowadzonego testu. Wynika to ze wspomnianego w rozdziale 4.1 kompromisu pomiędzy rozdzielczością czasową rozpoznawanych wysokości, a ich poprawnym dopasowaniem. Zastosowanie identycznych rozwiązań dla różnych instrumentów dało inną średnią skuteczność, co oznacza, że słusznym podejściem byłoby stworzenie dodatkowej klasy przechowującej takie ustawienia algorytmów, które dają najlepsze rezultaty dla konkretnych instrumentów. Czynnikami wpływającymi na ostateczną skuteczność algorytmu są przyjęte ustawienia:

- wyliczania współczynników MFCC (długość trwania ramki czasowej, zastosowana ilość filtrów melowych oraz ich zakres częstotliwości),
- tworzenia modeli GMM (ilość komponentów),
- długości analizowanych nagrań porównywanych z modelami GMM,
- algorytmu grupującego, który dzieli rozpoznane wysokości dźwięków na poprawne i niepoprawne.

Na podstawie rysunków 5.1 – 5.3 można zauważyć, że istniejące obecnie narzędzia do analizy wysokości dźwięków mogłyby posłużyć za fragment aplikacji, jednak bez dodatkowych funkcji nie są w stanie automatycznie segmentować nagrań. Domyślnie udostępniają możliwość wskazania na klasę wysokości dźwięków w krótkich chwilach czasowych lub określają prawdopodobieństwo wystąpienia danej wysokości w określonej oktawie.

Dalsze prace nad opisaną metodą segmentacji powinny skupić się na tym, aby dodać inne sposoby parametryzacji sygnału tak, by poza przyporządkowaniem wysokości dźwięków na podstawie opracowanych modeli sprawdzać dodatkowe, charakterystyczne cechy sygnału (np. częstotliwość podstawową), które miałyby wpływ na ostateczne wyniki.

Bibliografia

[1] Avery Li-chun Wang, An Industrial-Strength Audio Search Algorithm, Mat. Konferencyjne: 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, 26 – 30.10.2003

[2] Acoustical Society of America, dostępny: https://asastandards.org/ (odwiedzona 9.01.2019r.)

[3] Makarewicz R.: Dźwięki i fale. Poznań, Wydawnictwo Naukowe UAM 2004

[4] Pluta M.: Zasady muzyki i notacja muzyczna. Kraków, Wydawnictwo AGH 2012

[5] Mel Frequency Cepstral Coefficient (MFCC), dostępny: http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ (odwiedzona 17.12.2018).

[6] Douglas A. Reynolds, Richard C. Rose, Robust text-independent speaker identification using Gaussian mixture speaker models, IEEE Transactions on Speech and Audio Processing, 01.1995.

[7] J. VanderPlas, Python Data Science Handbook Essential Tools for Working with Data, O'Reilly Media, 11. 2016

[8] H. Akaike, Information theory and an extension of the maximum likelihood principle, Mat. Konferencyjne: Second International Symposium on Information Theory, Budapeszt, 1973.

[9] McFee B., Raffel C., Liang D., P.W. Ellis D., McVicar M., Battenberg E., Nieto O., librosa: Audio and music signal analysis in python, Mat. Konferencyjne: In Proceedings of the 14th python in science conference, 18-25. 2015.

[10] Lartillot O., Toiviainen P., A Matlab Toolbox for Musical Feature Extraction From Audio, Mat. Konferencyjne: International Conference on Digital Audio Effects, Bordeaux, 2007.

[11] G. Schwarz, Estimating the Dimension of a Model, The Annals of Statistics, 03.1978.