

Uniwersytet Mikołaja Kopernika w Toruniu

Wydział Matematyki i Informatyki

Patryk Bieszke

nr albumu: 273187

informatyka

Praca magisterska

# Analiza problemu transkrypcji muzyki i metod jego rozwiązań

Opiekun pracy dyplomowej  
prof. dr hab. Piotr Wiśniewski

Toruń 2019

# Spis treści

<b>Wstęp</b>	<b>3</b>
<b>1. Wprowadzenie do zasad działania muzyki</b>	<b>5</b>
1.1. Rytm . . . . .	7
1.2. Wysokość tonu . . . . .	8
1.3. Głośność . . . . .	10
1.4. Barwa dźwięku . . . . .	12
<b>2. Modelowanie systemu transkrypcji</b>	<b>14</b>
2.1. Reprezentacja danych średniego poziomu . . . . .	15
2.2. Transformata Fouriera . . . . .	16
2.2.1. Dyskretna transformata Fouriera . . . . .	18
2.2.2. Szybka transformacja Fouriera . . . . .	19
2.2.3. Krótkoczasowa transformata Fouriera . . . . .	20
2.2.4. Przetwarzanie transformacji Fouriera . . . . .	23
<b>3. Estymacja częstotliwości fundamentalnej w sygnale monofonicznym</b>	<b>25</b>
3.1. Funkcja autokorelacji . . . . .	25
3.2. Analiza TFR . . . . .	28
3.3. Analiza cepstralna . . . . .	30
3.4. ACLOS . . . . .	33
<b>4. Estymacja wielu częstotliwości fundamentalnych w sygnale polifonicznym</b>	<b>37</b>
4.1. Wyniki algorytmów przeznaczonych dla sygnałów monofonicznych . . . . .	38
4.2. Metody oparte na modelach generatywnych . . . . .	43
4.2.1. Przetwarzanie wstępne . . . . .	44
4.2.2. Modelowanie obwiedni spektrum . . . . .	45
4.2.3. Nieharmoniczność . . . . .	48
4.2.4. Metoda Pertusa i Iñesta (2008) . . . . .	49
4.2.5. Metoda Pertusa i Iñesta (2012) . . . . .	52
4.3. Metody z wykorzystaniem uczenia maszynowego . . . . .	55
4.3.1. Holistyczna metoda z użyciem sieci neuronowych . . . . .	56

4.3.2. Onsets and Frames . . . . .	58
<b>5. Implementacja</b>	<b>62</b>
5.1. Autokorelacja . . . . .	63
5.2. Cepstrum . . . . .	64
5.3. ACLOS . . . . .	66
5.4. Metoda Pertusa i Iñesta (2008) . . . . .	68
5.5. Metoda Pertusa i Iñesta (2012) . . . . .	69
5.6. Onsets and Frames . . . . .	71
5.7. Generowanie MIDI . . . . .	71
5.8. Porównywanie MIDI . . . . .	72
5.9. Interfejs GUI . . . . .	76
<b>6. Ewaluacja</b>	<b>78</b>
<b>Zakończenie</b>	<b>84</b>
<b>Spis rysunków</b>	<b>88</b>
<b>Spis tablic</b>	<b>89</b>
<b>Spis listingów</b>	<b>90</b>
<b>Bibliografia</b>	<b>91</b>

## **Wstęp**

Transkrypcja muzyki to proces polegający na zapisaniu danego utworu muzycznego w sposób formalny (jak np. zapis nutowy) z istniejącego zapisu dźwiękowego. Wynik tej operacji wykorzystywany jest do wiernego odtworzenia kompozycji źródłowych oraz lepszego zrozumienia muzyki, dając możliwość wygenerowania zbiorów danych transkrypcji z binarnych plików dźwiękowych. Zautomatyzowanie rozwiązania tego problemu nosi nazwę AMT (z ang. *Automatic Music Transcription*) i polega na utworzeniu transkrypcji (np. w postaci pliku MIDI) z pliku dźwiękowego.

Kompletna transkrypcja, czyli wykrycie wszystkich dźwięków wraz z ich pochodeniem i czasem występowania w sygnale muzycznym wymaga rozpoznania wysokości dźwięków, czasów ich występowania i rodzaju instrumentu, przy pomocy którego dane dźwięki zostały wygenerowane. Złożoność tego zadania jest bardzo wysoka, a w przypadku niektórych danych wejściowych uznawana za niemożliwą. Celem tak zdefiniowanej transkrypcji jest zatem wykrycie jak największej ilości składowych jak jest to możliwe. Terminem *częściowej transkrypcji* nazywa się znajdowanie dobrze zdefiniowanych części muzycznego sygnału, takich jak główna melodia i perkusja wraz z rytmem [28, s. 3–7].

Pierwsze próby zautomatyzowania procesu transkrypcji muzyki monofonicznej datowane są na lata sześćdziesiąte XX wieku. Bernard Gold w roku 1962 opublikował wypracowanie pod tytułem „Program komputerowy do wydobywania wysokości dźwięku” [13] (tłumaczenie własne, z ang. *Computer Program for Pitch Extraction*). Pierwsze próby transkrypcji muzyki polifonicznej odbyły się w latach siedemdziesiątych XX wieku, kiedy Moorer zaproponował system transkrypcji dwóch głosów w swojej pracy naukowej [34]. Wczesne prace naukowe nad tą tematyką dzielą ograniczenie liczby równoległych dźwięków do maksymalnie dwóch jednocześnie. Relacja pomiędzy wysokościami również miała duże obostrzenia w pionierskich odkryciach. Prace te były zorientowane na analizę tonu wokalnego w kontekście analizy mowy. Przykładem może być wcześniej przytoczona praca [13], która była napisana z myślą o urządzeniu zdolnym syntezować dźwięk mowy (tzw. vocoder). Metodologie używane do transkrypcji muzyki swoje fundamenty biorą właśnie z algorytmów analizujących mowę, które to były rozwijane w znacznie większym tempie w porównaniu do tych badających muzykę. Jak opisuje Anssi Klapuri w [28, s. 6], w pracy Goto i Muraoka z roku 1994 [14] po raz pierwszy pojawia się metoda do śledzenia rytmu w utworze muzycznym, natomiast pierwsze próby transkrypcji instrumentów perkusyjnych

odbyły się w latach osiemdziesiątych i dziewięćdziesiątych XX wieku w pracach naukowych [48, s. 99–104] i [3, s. 77–84].

Bez automatyzacji procesu, transkrypcja muzyki wykonywana manualnie polega na iteracyjnym odsłuchiwaniu danego utwóru, podczas którego ustalane są grane instrumenty wraz z częstotliwością fundamentalną lub akordem, który jest na nich grany. Ludzkie ucho postrzega dźwięk relatywnie, co powoduje, że dużo łatwiej jest porównywać dźwięki niż je analizować w odizolowaniu, toteż często wykorzystywane są dźwięki referencyjne.

Najnowsze dokonania w dziedzinie automatycznej transkrypcji muzyki są jednak nadal gorsze w poprawności rezultatów od transkrypcji manualnej profesjonalistów w dziedzinie muzyki. Rozwiązanie, które dawałoby rzetelne rezultaty w ogólnym zastosowaniu na muzyce jeszcze nie istnieje. Jednakże, z odpowiednimi założeniami co do sygnału źródłowego, najnowsze metody potrafią dokonać kompletnej transkrypcji utworu muzycznego. Restrykcje w analizowanym sygnale często sprowadzają się do ograniczenia polifoniczności (ilości jednoczesnych dźwięków) [9, s. 1–2], użycia tylko konkretnego instrumentu [17] czy braku interferencji z dźwiękami perkusyjnymi [21].

W tej pracy przeanalizowany jest problem automatycznej transkrypcji muzyki, nakreślając obecny stan nauki w tej dziedzinie. Omówione są najważniejsze rejony w oparciu o przykłady już istniejących rozwiązań jak i próba optymalizacji części z nich poprzez zrównoleglenie obliczeń z wykorzystaniem karty graficznej. Celem tych rozważań jest przybliżenie ogólnej koncepcji transkrypcji muzyki wraz z teorią na jakiej się opiera.

W rozdziale 1 zdefiniowane i omówione są pojęcia niezbędne do zrozumienia istoty problemu transkrypcji muzyki. Następnie, w rozdziale 2 omówione są ogólne założenia systemów transkrypcji muzyki, jak i przedstawione podstawowe narzędzia, z których korzystają. Rozdział 3 dotyczy metod wykrywania wysokości dźwięku sygnału monofonicznego. Te metody są w kolejnym rozdziale 4 omawiane w kontekście sygnałów polifonicznych. Rozdział ten zawiera również szczegółowy opis wybranych algorytmów dedykowanych do estymacji wysokości dźwięków w sygnałach polifonicznych. Szczegóły na temat implementacji omawianych algorytmów znajdują się w rozdziale 5. Przy użyciu tych implementacji zostało przeprowadzone porównanie rzetelności wyników jak i złożoności czasowej algorytmów, które opisane jest w rozdziale 6. Krótkie podsumowanie umieszczone jest na końcu pracy w zakończeniu.

## **1. Wprowadzenie do zasad działania muzyki**

W celu usystematyzowania pojęć i nadania fundamentu do dalszych rozważań istotne jest zdefiniowanie terminologii związanej z muzyką jak i samej muzyki. Utwór muzyczny jest uporządkowanym zbiorem dźwięków o odpowiednich wysokościach i głosnościach z zachowaniem odpowiedniego rytmu. Wysokości tych dźwięków są proporcjonalne do częstotliwości wibracji, które zostały wytworzone przez dany instrument lub głos ludzki. Głośność określana jest przy pomocy amplitudy (magnitudy) zadanego sygnału akustycznego. Sygnały monofoniczne odnoszą się do dźwięków, które nie nachodzą na siebie w czasie. Analogicznie, sygnały polifoniczne to takie, w których występuje więcej niż jeden dźwięk w tym samym momencie czasu. Starając się rozłożyć skomplikowaną strukturę utworów muzycznych na możliwie odseparowane części wyróżnia się cztery rozdzielne składowe, których zbadanie i poprawne zinterpretowanie jest przedmiotem algorytmów transkrypcji muzyki [22, s. 63]. Są to:

- Rytm
- Wysokość tonu
- Głośność
- Barwa dźwięku

Zapis nutowy utworu jest zdefiniowany w taki sposób, że wszystkie wyżej wymienione cechy są jednoznacznie opisane, z kolei ten jest wystarczającym źródłem informacji dla muzyków aby zagrać dany utwór muzyczny. Zapisane nuty przedstawiają wysokość tonu wraz z długością granego dźwięku i poziomem głośności, jaki powinien zostać zastosowany. Barwę dźwięku reprezentuje instrument muzyczny, do którego odnosi się dany zapis nutowy.

Znaki chromatyczne przykluczowe określają tonację (w notacji angielskiej zwana kluczem), w jakiej został napisany utwór. Opisuje ona harmoniczną podstawę kompozycji co czyni ją bardzo istotną informacją, która dostępna jest w zapisie nutowym. W praktyce klucz ogranicza dozwolone wysokości tonu, jakie mogą być zagrane. Wiele kompozycji muzycznych dynamicznie manipuluje kluczem, tworząc mniej statyczne linie melodyczne, co uniemożliwia opieranie transkrypcji o ten parametr.

Zapis nutowy zawiera też informacje o tempie utworu, czyli szybkość z jaką kolejne dźwięki powinny być grane, jak i sygnaturę czasową, która wyznacza bazową

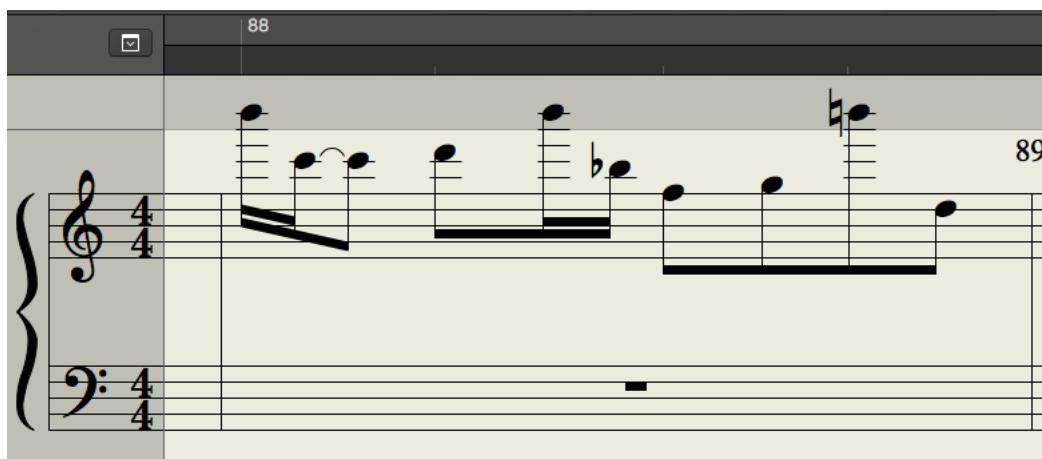
strukturę rytmiczną w jakiej dany utwór został napisany. Te dwie informacje są częścią wylistowanego wyżej rytmu.

Barwa dźwięku jest ścisłe związana z instrumentem, którego dotyczy dany zapis nutowy. Każde ze źródeł dźwięku cechuje się odrębna barwą, nawet gdy uwzględnić taką samą wysokość i głośność sygnału fonicznego. Problem rozróżniania tej cechy jest związany nie tylko z transkrypcją muzyki, ale i algorytmami identyfikującymi instrumenty poprzez analizę barwy dźwięku.

Kompletna transkrypcja muzyczna jest złożonym zadaniem, które ze względu na specyfikę fal akustycznych można podzielić na trzy pod-zadania:

- rozpoznawanie elementów perkusyjnych razem ze śledzeniem rytmiki na podstawie rytmu i głośności (1.1, 1.3),
- wykrywanie częstotliwości fundamentalnych na podstawie wysokości tonu (1.2),
- kategoryzowanie instrumentów występujących w sygnale na podstawie barwy dźwięku (1.4).

W dalszej części tego rozdziału zostaną dokładniej opisane cztery wymienione wyżej cechy sygnału akustycznego.



*Rysunek 1.1: Przykład notacji muzycznej wygenerowanej na podstawie pliku MIDI w programie Logic Pro X. Widać na nim takie informacje jak metrum czy wysokości i długości granych dźwięków*

## 1.1. Rytm

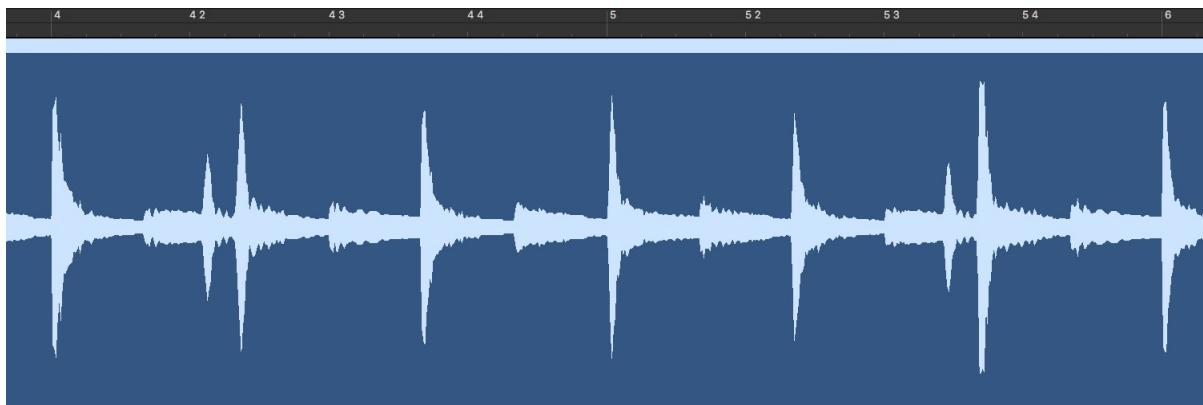
W kontekście muzykologii, rytm jest relacją czasu pomiędzy poszczególnymi dźwiękami, tworzącą strukturę rytmiczną utworu. Nie można niestety zakładać stałości rytmu na przestrzeni całej kompozycji, ponieważ powszechną praktyką kompozytorów jest jego modyfikacja na przestrzeni piosenki w celu uzyskania konkretnych efektów artystycznych. Dla przykładu, minimalne podnoszenie tempa na czas każdego z refrenów jest często stosowane w muzyce popularnej.

Informacja o rytmie jest zapisana w relacjach pomiędzy poszczególnymi dźwiękami i pomiędzy większymi strukturami wewnątrz utworu. Odstęp pomiędzy początkami dźwięków (często określany jako IOI, z ang. *Inter-onset interval*) pełni bardzo istotną rolę w rytmice utworu, jak i strukturze perkusyjnej. Pojęcie to odnosi się do interwału czasowego pomiędzy dwoma początkami dźwięków (np. moment uderzenia w werbel), która ta cecha jest dużo bardziej istotna w rytmice niż długość danych dźwięków w sekundach [7, s. 482–489].

Podstawową jednostką rytmiczną jest nuta. Miarą tempa utworu przeniesioną na domenę fizyczną są uderzenia na minutę (często określane jako BPM, z ang. *beats per minute*) oznaczające liczbę ćwierćnutek (beatów) przypadających na jedną minutę. Taktem nazywa się odcinek zapisu muzycznego wizualnie oznaczony pionowymi kreskami. Takty tworzą regularne odcinki czasowe, a ich długość jest zależna od metrum. Metrum jest oznaczeniem metrycznym opisującym ile jakich jednostek mieści się w takcie. Dla przykładu, najpopularniejszym metrum w muzyce popularnej jest 4/4, co oznacza, że w każdym taktie mieszą się cztery ćwierćnuty.

Zbadanie rytmiki wymaga wyznaczenia metrum i długości poszczególnych jednostek metrycznych utworu. Ze względu na to, że nagrania muzyczne cechują się periodycznością wokół tych jednostek, potrzebne do tego zadania jest wykrycie wzorca regularnych, przeplatanych akcentowanych i nieakcentowanych uderzeń (często nazywanych słabszymi i silniejszymi) [31, s. 12–35]. Pozyskane *pulsy* o różnych czasowych poziomach tworzą razem strukturę rytmiczną. Przykład takiej struktury przedstawiony jest na rysunku 1.2, który przedstawia sygnał nagranej perkusji. W równych odstępach pojawiające się piki oznaczają stopkę, werbel oraz talerze, każde występujące okresowo.

W skład rytmu wchodzi również grupowanie. Termin ten odwołuje się do powiązań pomiędzy pojedyńczymi dźwiękami, które razem tworzą łączne frazy, a te z kolei



Rysunek 1.2: Sygnał akustyczny z wyraźnymi metrykami muzycznymi. Na osi poziomej liczbowo oznaczona jest metryka nutowa (4 beaty), mniejszymi zaś regularne odstępy tak-towe.

dalej grupowane tworzą większe i bardziej złożone jednostki w hierarchicznej strukturze [31, s. 12–35].

## 1.2. Wysokość tonu

Wysokość tonu, czy też wysokość dźwięku, jest percepcyjnym atrybutem za pomocą którego porządkowane są dźwięki na skali częstotliwości, od najniższych do najwyższych. Jest to pojęcie tożsame z częstotliwością sygnału sinusoidalnego zgodnego z tym, które jest rejestrowane przez ludzkie ucho [15, s. 3492–3493].



Rysunek 1.3: Ilustracja przedstawiająca trzy oktawy klawiatury pianina z oznaczonymi muzycznymi nazwami nut

Częstotliwość fundamentalna (w dalszej części oznaczana jako  $F_0$ ), jest terminem fizycznym dla sygnałów okresowych i prawie okresowych, zdefiniowanym jako odwrotność długości okresu sygnału. W niejednoznacznych przypadkach  $F_0$  określony jest jako okres odpowiadający postrzeganej wysokości tonu [28, s. 8].

F0 jest ściśle powiązana z wysokością tonu. Współczesna muzyka zachodnia opiera się na skali melowej. System równomiernie poddaje nuty kwantyzacji na logarytmicznej skali spektrum częstotliwości na tak zwane półtony, czyli najmniejsze interwały. Każda z tak powstałych muzycznych nut ma przypisaną literę w celu odróżnienia, alfabetycznie od A do G razem z podwyższeniami (moll, w języku angielski *sharp*, oznaczany symbolem ♯) i obniżeniami (dur, w języku angielskim *flat*, oznaczany symbolem ♭) półtonowymi. Tak nazwane wysokości są grupowane w oktawy, z których każda zawierająca 12 półtonów. Pełna nazwa granego dźwięku powinna zawierać nazwę literalną wraz z numerem oktawy. Ze względu na tak skonstruowany system nazewnictwa, niektóre z dźwięków mogą mieć dwie poprawne nazwy (np. A♯4 i B♭4). O tym, którą z tych dwóch nazw należy użyć, zazwyczaj zależy tonacja w jakiej utwór został napisany [5, s. 215–220].

Omówiony system zakłada referencyjne mierzenie częstotliwości zależnie od wysokości A4, która powinna być równa 440 Hz. Każde podwyższenie danej nuty o oktawę powoduje dwukrotne zwiększenie się częstotliwości podniesionego dźwięku. Standard MIDI [23, s. 67–71] mapuje daną częstotliwość bazową na liczbę rzeczywistą odpowiadającą wysokości dźwięku przy pomocy wzoru:

$$p = 69 + 12 * \log_2\left(\frac{f}{440Hz}\right) \quad (1.1)$$

Istnieją oczywiście instrumenty, które generują dźwięki o dowolnej częstotliwości fundamentalnej, a nie jedynie o dyskretnej wartości, jak na przykład skrzypce czy saksofon. Kwantyzacja wysokości dźwięku jest procesem, który kategoryzuje dźwięki do najbliższej wartości w sensie częstotliwości nazwanych nut, dzięki czemu można je formalnie zapisać w procesie transkrypcji.

Interesującą własnością niemalże wszystkich kultur muzycznych jest podobieństwo do siebie odpowiednich nut w oktawach. Nuty A2, A3 i A4 dla przykładu mają taką samą rolę w kontekście harmoniczności. Brzmią one bardzo podobnie do siebie, jedynie różnią się wysokością. Ze względu na tą cechę, opisany wyżej system redukuje liczbę wszystkich istniejących dźwięków do jedynie 12 klas dźwięków, po 1 na każdą z nut w oktawie. Korzystając ze wzoru 1.1 dla wielokrotności liczby 440 można uzyskać tabelkę z częstotliwościami dźwięków *A* w kolejnych oktawach, jak w tabeli 1.1.

Relacja pomiędzy wysokościami używanych tonów jest ważną informacją przy analizie muzycznej. Mimo dużej ilości możliwych częstotliwości, używane nuty często ograniczają się do 12 klas dźwięków.

p	f	Nazwa wysokości dźwięku
33	55	A1
45	110	A2
57	220	A3
69	440	A4
81	880	A5
93	1760	A6
105	3520	A7

Tablica 1.1: Wysokości kolejnych dźwięków A zgodna ze wzorem 1.1

niczane są do podzbiorów zależnych od tonacji kompozycji. Badając rozkład częstotliwości takie informacje mogą znacznie ograniczyć błędy podczas analizy sygnału. Należy jednak mieć na uwadze fakt, że nagrany instrument może nie posiadać dokładnie takiej samej częstotliwości, jak jest oczekiwana dla danej wysokości tonu. Dzieje się tak ze względu na charakterystykę instrumentu czy też techniczne parametry nagrania. Dodatkowym utrudnieniem są różne stroje muzyczne, które inaczej przyporządkowują częstotliwości do nut. W tej pracy badane są jedynie utwory bazujące na wyżej opisanyem systemie (zwanyem system równomiernie temperowanym), jednak na uwadze należy mieć fakt, że nie jest to uniwersalne rozwiązanie. Również z artystycznych względów docelowe częstotliwości generowane przez instrumenty odbiegają od tych przewidzianych przez opisany system. Maksymalną nieharmoniczność (odchylenie od prawidłowego F0) najczęściej ustala się na poziomie 5%, lecz nie istnieje sformalizowany dozwolony zakres. Próba rozpoznawania jedynie konkretnych częstotliwości tonów jest dużym uproszczeniem w analizie sygnałów fonicznych, jednocześnie znacząco ograniczającym jej uniwersalność [22, s. 64–65] [28, s. 7–11].

### 1.3. Głośność

Termin głośności z reguły nie odnosi się do pojedynczych nut, ale do większych sekcji. W notacji muzycznej nie często używa się słowa „głośność”. To, co słuchacz odbiera jako głośność dźwięku z punktu widzenia muzyka jest dynamiką granego fragmentu.

Większość instrumentów używanych do tworzenia muzyki posiada zdolność wytwarzania zróżnicowanych w sile fal akustycznych. Podstawą skali dynamiki są dwa słowa kluczowe:

- **f** (forte) - głośno
- **p** (piano) - cicho

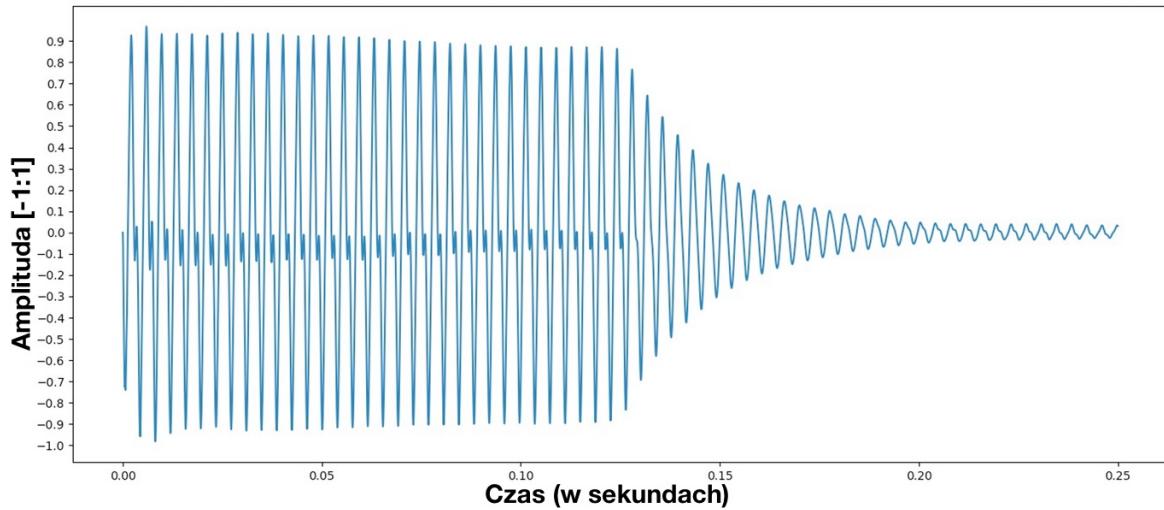
Pomiędzy wymienionymi stopniami istnieją kroki pośrednie, odpowiednio mezzo-forte (dość głośno) i mezzo-piano (dość cicho). Poza czterema wymienionymi przypadkami dostępne są również skrajności odpowiadające za bardzo głośny (odpowiednio - cichy) i możliwie najgłośniejszy (odpowiednio - najcichszy) odcień dynamiki.

Z punktu widzenia algorytmicznej analizy muzyki problematyczne są dynamiki zwane crescendo i diminuendo. Odpowiadają one za kolejno stopniowe wzmacnianie i stopniowe osłabianie natężenia dynamiki. Łatwo jest pomylić takie zabiegi z innymi cechami sygnału dźwiękowego.

Warto też wspomnieć o technice zwanej tremolo, która określa artykulacje grania wielu dźwięków o tej samej wysokości z różną dynamiką. Analizując sygnał łatwo jest pomylić kolejne uderzenia tremolo z zaganiem nowych nut.

Z praktycznego punktu widzenia głośność w plikach muzycznych jest zdeterminowana przez to, jak dany utwór został wyprodukowany. W procesie nagrywania ogromne znaczenia ma położenie i rodzaj mikrofonu, jak i otoczenie w którym rejestrowany jest dźwięk. Mikrofony posiadają różne pasma przenoszenia i charakterystyki kierunkowe, które nie tylko mogą zniekształcić głośność nagrywanego instrumentu, ale również jego barwę. Różne rodzaje mikrofonów są wykorzystywane przy różnych źródłach dźwięku, np. mikrofony pojemościowe są odpowiedniesze do nagrywania wokali i instrumentów akustycznych niż wstęgowe, ponieważ przenoszą częstotliwości tych właśnie źródeł bardziej wiarygodnie [22, s. 48–52]. Charakterystyka niektórych powierzchni może sprawić, że oryginalnie cichsze częstotliwości będą wychodziły na przód nagrania. Mocna kompresja dynamiki utworu jest powszechną praktyką podczas procesu mikowania i masterowania utworów. Dzieje się tak, ponieważ ludzkie ucho uważa głośniejsze utwory za lepsze, co doprowadziło rynek muzyczny do produkowania utworów o niemal stałej dynamice głośności, co w skrajnych przypadkach powoduje upodobnianie się krzywej sygnału akustycznego do białego szumu. Choć temat ten jest w trakcie regulacji, między innymi poprzez wprowadzenie ograniczeń dynamiki na podstawie jednostki LUFS danego utworu, dokładne

analizowanie głośności wielu istniejących nagrań może doprowadzić do nieprawdziwych wniosków.



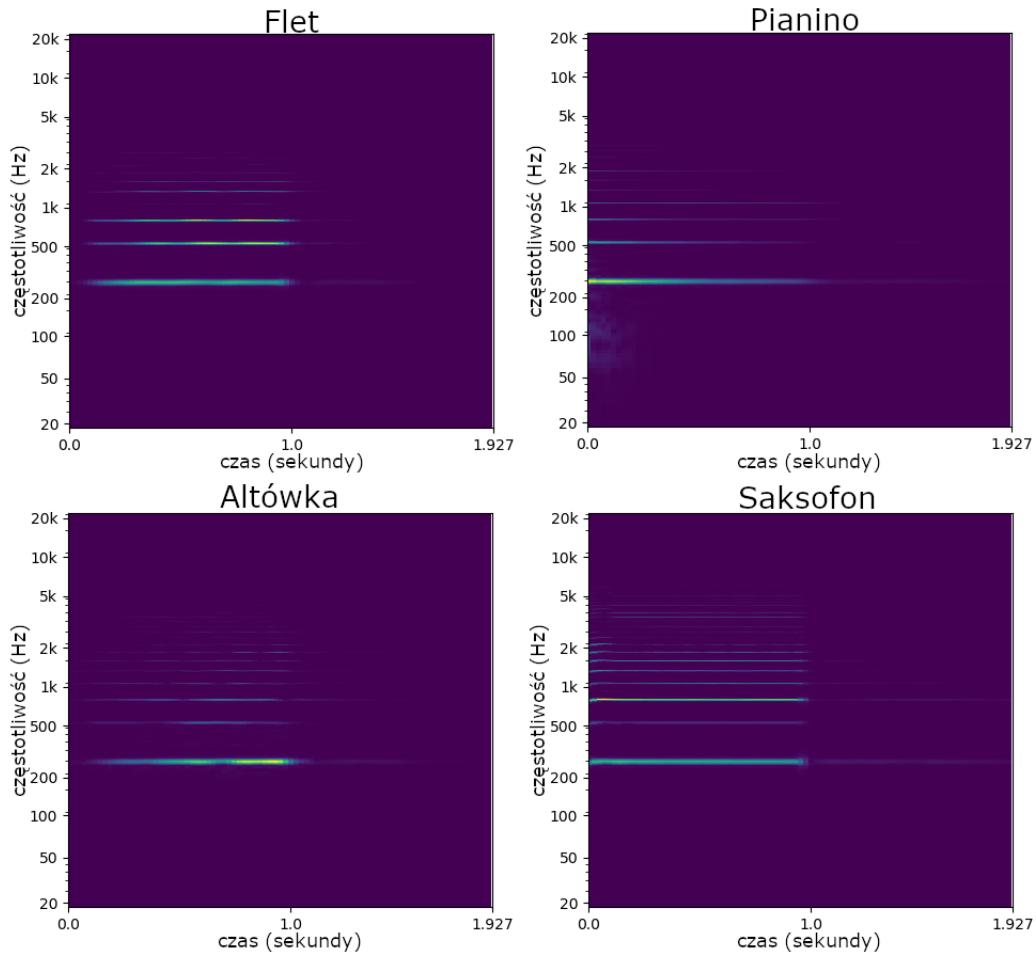
Rysunek 1.4: Wykres przedstawiający sygnał dźwięku o zmiennej amplitudzie (głośności) względem czasu wygenerowany przez klasyczne pianino elektryczne.

## 1.4. Barwa dźwięku

Barwa dźwięku (inaczej tembr) jest cechą odwołującą się do harmonicznej domeny sygnału fonicznego. Pozwala ona na odróżnienie od siebie dźwięków o tej samej głośności i wysokości granych na różnych instrumentach. Silne tony harmoniczne danej barwy sprawiają, że dźwięk jest bardziej wyrazisty. W przeciwnym wypadku, gdy siła tonów harmonicznych jest mała, sygnał staje się bardziej rozmyty dla odbiorcy. Barwa dźwięku jest zależna od amplitudy poszczególnych tonów harmonicznych, ich rozkładu w widmie jak i samej struktury tego widma. Jest to wyraźnie widoczne na rysunku 1.5.

Ułożenie tonów harmonicznych jest zintegrowane z harmonią pomiędzy różnymi granymi nutami. Gdy dwa dźwięki są grane w tym samym czasie, nakładające się na siebie harmoniczne sprawiają, że dźwięk brzmi pozornie spójnie. Generuje to dodatkowe komplikacje dla algorytmów wykrywających wysokość tonów, które muszą odseparować nakładające się na siebie nuty.

Barwa dźwięku nie jest stała w czasie. Harmoniczne o wysokich częstotliwościach szybciej cichną w porównaniu do tych o niższych. Powoduje to, że nie tylko średnia głośność jest mniejsza, ale także barwa ulega modyfikacji. Dla dźwięków instrumentów dętych czy skrzypiec cechy jak wysokość tonu, głośność i barwa pozostają relatywnie



Rysunek 1.5: Spektrogramy dźwięku C3 (130.81Hz) grane na czterech różnych instrumentach (flet, pianino, altówka i saksofon) pokazują różnice w ilości i rozmieszczeniu harmonicznych każdego z dźwięków. Spektrogramy wykonane z 4096 samplami na okno, odstępem okien 1024 sampli i funkcją okna Hanna. Jaśniejsze obszary symbolizują większą magnitudę.

stałe przez cały okres grania nuty, natomiast dźwięki pianina „rozmywają się” w czasie przez ich zmienną barwę. Ta cecha barwy dźwięku musi zostać zaadresowana przy próbie dokładnej analizy sygnału fonicznego [22, s. 64–65] [27, 804–805].

Istotnym pojęciem przy omawianiu barwy dźwięku jest formant. Formantem nazywane jest pasmo, w którego obrębie częstotliwości harmoniczne są wzmacnione względem pozostałych w danym sygnale. Formanty mogą sprawić, że harmoniczne będą miały większą amplitudę niż częstotliwość fundamentalna. Istnienie tego zjawiska fonicznego uniemożliwia poprawność podejścia, w którym jako częstotliwość fundamentalną wybierana byłaby ta, o największej amplitudzie w analizowanym oknie [25, s. 24–25].

## 2. Modelowanie systemu transkrypcji

Sygnały akustyczne, przetworzone przez przetwornik analogowo-cyfrowy (A/C) są przechowywane w postaci binarnych plików dźwiękowych jako dyskretne sygnały cyfrowe dyskretnego czasu. W procesie cyfryzacji zostają poddane procesowi kwantyzowania wartości, najczęściej poprzez zaokrąglenie do najbliższych liczb całkowitych [54, s. 1–4]. Sposób, w jaki pliki te przechowują dane, jest zdeterminowany przez wiele parametrów, takich jak kodowanie, które zostało na nich zastosowane, częstotliwość próbkowania, ilość kanałów itp. Istnieje wiele kodowań przeznaczonych do danych dźwiękowych, zarówno nieskompresowanych jak i skompresowanych. Te drugie powstały z myślą o zaoszczędzeniu jak największej ilości pamięci w zależności od tego, w jakim stopniu można obniżyć jakość podczas kompresji sygnału [22, s. 66]. Częstotliwość próbkowania ma również znaczący wpływ na wynikowy rozmiar plików, jak i ilość bitów przeznaczony na pojedyńczy sampel. Wszystkie cechy, jakie posiada cyfrowa reprezentacja rzeczywistego dźwięku, muszą być wzięte pod uwagę podczas jej przetwarzania.

W kontekście algorytmów generujących muzykę, binarne pliki dźwiękowe nie posiadają efektywnej struktury pozwalającej na trzymanie akustycznych informacji o melodii czy rytmice. Cechy, które można wydobyć ze zdigitalizowanego strumienia audio mogą posłużyć za istotne wskaźniki przy klasyfikacji zbiorów, natomiast kluczowe informacje odnoszące się do struktury kompozycji i harmonii poszczególnych sekcji szukać należy w transkrypcji danego utworu.

Utwór muzyczny posiada bardzo wiele własności, na podstawie których można opierać analizę. W celu zoptymalizowania algorytmu automatycznej transkrypcji należy przefiltrować wszystkie te dane i zdecydować się, które uznać za istotne, a które za zaniedbywalne. Strukturyzacja algorytmu i dobór reprezentacji danych są również istotne przy projektowaniu algorytmu.

Neuropsychologiczne badania, które były prowadzone w kontekście przetwarzania muzyki przez ludzki umysł dowodzą, że analiza dźwięku jest dzielona na pod-zadania. Jak opisują autorzy w [53] pewne własności muzyki są przez mózg odseparowywane od reszty bez szerszego kontekstu. Daje to informacje o tym, że modularyzacja podejścia do transkrypcji może być właściwym rozwiązaniem tego problemu. W automatycznej transkrypcji muzyki często używa się innych reprezentacji danych do analizy wysokości dźwięku, w których większy nacisk kładzie się na wierne odwzorowanie częstotliwości, a

innych do analizy rytmu, w których najważniejsze jest dokładność w domenie czasu [28, s. 11–13].

W pracy naukowej [1, s. 22–24] autorzy opisują obecny stan nauki w dziedzinie transkrypcji muzyki i kategoryzują oni algorytmy AMT ze względu na poziom abstrakcji używanych struktur na cztery kategorie:

- transkrypcje na poziomie okien czasowych, badające każde okno osobno w poszukiwaniu kandydatów F0. Algorytmy na tym poziomie wykorzystują między innymi tradycyjne metodyki przetwarzania sygnału, modelowanie probabilistyczne, podejścia bayesowskie, użycie nieujemnej faktoryzacji macierzy (NMF, z ang. *non-negative matrix factorization*) i sieci neuronowe (NN, z ang. *Neural Network*). Ta praca skupia się na tradycyjnym przetwarzaniu sygnału z elementami podejścia bayesowskiego (3, 4.2.4) ale opisana jest również sieć neuronowa wykazująca najlepsze do tej pory rezultaty (4.3.2),
- transkrypcje na poziomie nut, zwane też śledzeniem wysokości. W odróżnieniu od poziomu okien czasowych, w tym podejściu analizowane są sąsiednie wyniki w celu połączenia ich w spójne muzycznie całości. Przykład tego podejścia opisany jest w 4.2.5,
- transkrypcje na poziomie strumienia, zwana także strumienowaniem wielotonowym (MPS, z ang. *Multi-Pitch Streaming*), które grupuje nuty wydobyte z poziomu nut w strumieniu, najczęściej odpowiadające charakterystyce źródła fali akustycznej (odpowiadające instrumentowi),
- transkrypcje na poziomie notacji muzycznych. Jest to poziom najmniej zbadany na ten moment. Jego celem jest wygenerowanie transkrypcji w postaci czytelnej dla człowieka. Poziom ten zakłada wydobycie informacji o strukturach muzycznych, które nie są istotą analizy pozostałych algorytmów. Przykład takiej analizy jest opisany w 4.3.1 przy użyciu sieci neuronowych.

### 2.1. Reprezentacja danych średniego poziomu

Reprezentacja danych średniego poziomu pozwala na utworzenie pewnego rodzaju pośrednika pomiędzy akustyczną reprezentacją dźwięku a tą formalnie zapisaną. Sygnał muzyczny w czystej postaci jest mało informatywny, bo nie można z niego wyczytać chociażby poszczególnych nut utworu, tak jak jest to widoczne w zapisie nutowym.

W książce [28, s. 13–17] autorzy sugerują, że celem użycia tego narzędzia jest stworzenie pośredniego stopnia abstrakcji, które będzie działało jak interfejs do analizy audio jednocześnie ułatwiając konstrukcję systemów transkrypcji.

Najczęściej używaną reprezentacją danych na średnim poziomie przy analizie sygnału akustycznego jest szybka transformacja Fouriera (dokładniej opisana w sekcji 2.2.2) sygnału w kolejnych oknach czasowych. W ogólności wszystkie funkcje pozwalające na przeniesienie sygnału z domeny czasowej na domenę częstotliwości są niezwykle istotne w analizie dźwięku. W tej pracy znajduje się opis algorytmów operujących na tej właśnie reprezentacji danych (analiza tej reprezentacji opisana jest w sekcji 3.2, algorytmy jak ACLOS opisany w sekcji 3.4 czy cepstrum opisane w 3.3 działają jedynie na widmie sygnału).

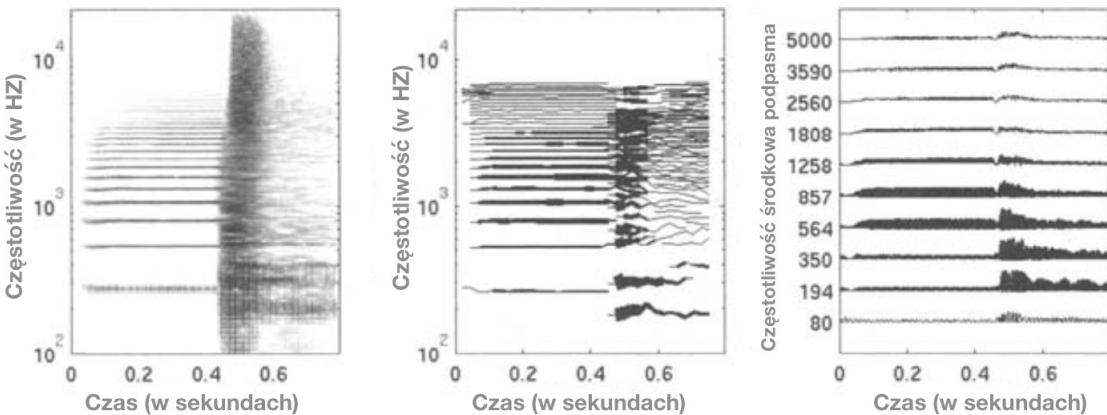
W książce [28, s. 13–17] autorzy uważają, że powszechnym wyborem dla reprezentacji danych na średnim poziomie jest bazowanie na *ścieżkach sinusoidalnych*. W tym przypadku sygnał akustyczny jest przedstawiony jako suma sinusoid o zmiennych w czasie częstotliwościach i amplitudach. Korzyścią, jaka płynie z tej reprezentacji jest łatwe przedstawienie instrumentów tonalnych. Podejście to nie sprawdza się jednak przy wielotonalnych sygnałach, gdzie dźwięki nakładają się na siebie w domenie częstotliwości i czasu.

Modele bazujące na słuchu człowieka są również używane jako reprezentacje danych na średnim poziomie. Sposób, w jaki ludzki mózg przetwarza muzykę, jest w zasadzie tym, czym jest istota transkrypcji, więc naturalnym wydaje się próba naśladowania procesów neuronowych. Sposób ten ma podłożę neuropsychologiczne. Dokładne opisy modeli słuchowych znajdują się między innymi w pracach [49] czy [53].

Nie ma jednej reguły, która decydowałaby o trafności reprezentacji danych na średnim poziomie. Dobór metody powinien być świadomy, zgodny z wymaganiami systemu transkrypcji, w którym reprezentacja ma zostać użyta [28, s. 13–15]. Różnice w reprezentacjach można zaobserwować na wykresach na rysunku 2.1.

## 2.2. Transformata Fouriera

Pliki dźwiękowe zawierają informacje, które reprezentują siłę ciśnienia akustycznego w domenie czasu. Jest to jedna z reprezentacji, która może być analizowana w celu zbadania konkretnych cech sygnału, lecz jest ona bezpośrednim efektem nagrania muzyki



Rysunek 2.1: Grafy przedstawiają 3 różne reprezentacje danych na średnim poziomie tego samego dźwięku trąbki grającej dźwięk C4. Lewy wykres przedstawia spektrogram na logarytmicznej skali częstotliwości. Środkowy wykres przedstawia reprezentację ścieżek sinusoidalnych, przy czym szerokość lini oznacza amplitudę poszczególnych sinusoid. Prawy wykres przedstawia prosty model słuchowy. Wykres z [28, s. 14], tłumaczenie własne.

i możliwości jej bezpośredniego odtworzenia. Dokładna analiza relacji sygnału muzycznego, jego reprezentacja jak i jego zrozumienie odbywa się po przeniesieniu na płaszczyznę częstotliwości. W muzyce zachodniej, F0 w zapisie nutowym definiuje się jako wysokość dźwięku w rozumieniu częstotliwości, co zostało omówione w sekcji 1.2.

Aby uzyskać reprezentację w domenie częstotliwości z sygnału akustycznego w domenie czasu powszechnie używane jest narzędzie matematyczne o nazwie transformata Fouriera. Funkcja ta jest uniwersalna i powszechnie stosowana w analizie sygnałów, ta praca skupia się jednak na opisaniu sposobów wykorzystywania jej do celów analizy sygnałów akustycznych. Transformata ta opiera się na założeniu, że każdą okresową krzywą można zapisać w postaci sumy składowych sinusoidalnych o różnych częstotliwościach. Funkcja ta przekształca sygnału z dziedziny czasu na dziedzinę częstotliwości, co daje zarys tego jak rozłożona jest amplituda poszczególnych częstotliwości w analizowanych danych. Transformata zadana jest wzorem:

$$CFT_x(f) = \int_{-\infty}^{\infty} f(t) e^{-j2\pi ft} dt \quad (2.1)$$

Gdzie  $F(f)$  nazywane jest transformatą Fouriera  $f(t)$  lub widmem częstotliwościowym. Wynik transformaty jest wektor liczb zespolonych, dzięki czemu można ją odwrócić przy

pomocy odwrotnej transformaty Fouriera (IFT, ang. *Inverse Fourier Transformation*) opisanej wzorem:

$$ICFT_x(f) = \int_{-\infty}^{\infty} F(t) e^{j2\pi ft} dt \quad (2.2)$$

[28, s. 22].

Do analizy całego spektrum utworu muzycznego, pojedyńcza transformata Fouriera może nie być wystarczająca ze względu na długość i zróżnicowanie utworu względem czasu. Reprezentacja czasu-częstotliwości pozwala na przedstawienie sygnału w postaci zespolonej w domenie czasu i częstotliwości (ang. time–frequency representation, TFR). W celu uzyskania takiej reprezentacji przy pomocy FT konieczne jest zastosowanie okna czasowego, co jest szerzej opisane w sekcji 2.2.3. Okna te pełnią rolę prostego TFR, spektrogramu jak i reprezentację czasową czy reprezentację opóźnienia czasowego [54, s. 22–23][42, s. 26–29] [28, s. 22–25].

### 2.2.1. Dyskretna transformata Fouriera

Jednym z założeń transformacji Fouriera jest, aby  $f(t)$  z 2.1 była funkcją ciągłą na przedziale od minus nieskończoności do nieskończoności, co przekłada się na transformatę w tym właśnie zbiorze. Z praktycznego punktu widzenia nieskończone granice sumy są niemożliwe do zrealizowania w analizie sygnału akustycznego, ponieważ przetwarzane są tylko skończone długości krzywych, co powoduje, że granice są zawsze skończone. W celu analizy na sygnałach, zwłaszcza w kontekście transkrypcji muzyki, danebrane pod uwagę sąbrane z ograniczonego przedziału czasowego. Potrzebne do tego celu będzie podzielenie analizowanych danych na krótkie próbki [42, s. 29–30][41].

Dyskretna transformata Fouriera (DFT) jest narzędziem matematycznym analogicznym do podstawowej transformaty Fouriera, lecz przeznaczonym do analizowania sygnału dyskretnego  $x(n)$  o okresie  $N$ , opisanego następującym wzorem:

$$x(n) = x(n + mN) \quad (2.3)$$

Pozwala ona na analizę pojedyńczego okresu zadanego sygnału. Własność ta jest szczególnie przydatna przy komputerowej analizie sygnałów, w tym fonicznych. Jest to transformacja zawsze odwracalna.

Założeniem DFT jest wyznaczenie iloczynu sygnału i okna prostokątnego (które opisane jest w sekcji 2.2.3), które to jest wycinkiem krzywej sygnału podlegającym ana-

lizie. W związku z cyklicznością zakładanego sygnału (2.3) przyjmuje się, że analizowany sygnał jest okresowy z okresem równym  $N$ , co implikuje obliczenie dokładnie  $N$  składowych harmonicznych analizowanych danych ( $N$  różnych częstotliwości). Dyskretny czas, częstotliwość i skończona liczba próbek zadanego sygnału sprowadza omawianą transformację do funkcji w pełni dyskretnej. Wynik dyskretyzacji TF jest następujące równanie:

$$DFT_x(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn}, k = 0, 1, 2, \dots, N - 1 \quad (2.4)$$

DFT, tak samo jak FT jest odwracalna, co można opisać wzorem:

$$IDFT_x(n) = \sum_{k=0}^{N-1} X(k)e^{j2\pi kn}, n = 0, 1, 2, \dots, N - 1 \quad (2.5)$$

Wynikiem transformacji będzie  $N$  liczb, odpowiadających kolejno amplitudzie i fazie sinusoid od  $-\frac{1}{2}\Delta$  do  $\frac{1}{2}\Delta$ , gdzie  $\Delta$  jest częstotliwością próbkowania, w rozdzielczości  $\frac{1}{N}\Delta$ . Częstotliwości są ustalone przez częstotliwość próbkowania oraz liczbę próbek w oknie. Jak opisuje Tomasz Zieliński w [54, s. 198 – 200, 204–206], DFT stanowi pewną aproksymację  $N$ -elementowego wektora próbek analizowanego sygnału za pomocą  $N$  ortogonalnych wektorów bazowych  $N$ -elementowych, z wyjątkiem sytuacji, kiedy w sygnale występuje składowa sinusoidalna, która nie jest częścią zbioru wektorów bazowych, bo w takim wypadku zostanie ona przedstawiona jako suma większej liczby sygnałów „bazowych”, co oznacza, że jej widmo ulegnie rozmyciu.

### 2.2.2. Szybka transformacja Fouriera

DFT omawiana w sekcji 2.2.1 jest wystarczająca do wykonania wszelkich analitycznych przekształceń. Problem jednak pojawia się w praktyce, i jest nim efektywność algorytmu. W swojej podstawowej formie transformata Fouriera  $F_n$  wymaga zliczenia wszystkich wartości, od  $h_0$  do  $h_{N-1}$ , dla każdej z częstotliwości  $n$ , co oznacza, że złożoność obliczeniowa dla  $N$  danych wynosi  $O(N^2)$ . Jest to zbyt duże obciążenie nawet dla najlepszych pod względem mocy obliczeniowej współczesnych komputerów do analizy dużego zbioru danych. Istnieje wiele metod optymalizacji numerycznych równań 2.4 i 2.5 zmniejszających złożoność obliczeniową do co najwyżej  $O(N \log_2 N)$ . Algorytmy te nazywają się szybkimi transformacjami Fouriera (FFT, z ang. *Fast Fourier Transform*) i to z nich korzysta się powszechnie podczas cyfrowej analizie sygnałów [8].

Istnieje wiele algorytmów zaliczających się do grupy algorytmów FFT. Popularnym algorymem jest algorytm *Cooley-Tukeya* w formie *Radix-2* (FFT o podstawie 2).

Korzysta on z metodologii dziel i zwycięzaj, dzieląc próbki danych wielkości  $N$  na dwie oddzielne przeplatane grupy wielkości  $N_1$  i  $N_2$  ( $N_1 + N_2 = N$ ) o indeksach odpowiednio parzystych ( $0, 2, 4, \dots$ ) i nieparzystych ( $1, 3, 5, \dots$ ) wykonując DFT na każdym ze zbiorów, a następnie odtwarza kompletne widmo sygnału z otrzymanych wyników częstkowych. W uproszczeniu algorytm wygląda następująco:

1. Póbki dzielone są rekurencyjnie na próbki o indeksach parzystych i nieparzystych tworząc przeplatane ciągi, aż do uzyskania zbiorów dwuelementowych,
2. Na każdym z uzyskanych zbiorów wykonuje się DFT, czyli łącznie  $\frac{N}{2}$  dwupunktowych DFT,
3. Proces składania widm powtarzany jest tak, że dwupräżkowe widma składane są w czteropräżkowe, czteropräżkowe w ośmiopräżkowe itd., do momentu, w którym od- tworzono zostanie widmo  $N$ -präżkowe, czyli kompletne widmo sygnału.

Co daje łączną ilość etapów operacji równą  $\log_2 N$  [54, s. 241–252].

### 2.2.3. Krótkoczasowa transformata Fouriera

Do tej pory przeanalizowane zostało działanie i podstawowe odmiany algorytmu transformacji Fouriera, jednak aby można było wykorzystać to narzędzie przy analizie utworów muzycznych niezbędne jest wprowadzenie reprezentacji czasu-częstotliwości TFR. Krótkoczasowa transformacja Fouriera (STFT) jest przeznaczona do operowania na małym oknie czasowym analizowanego sygnału. Pojedyncze okno o indeksie  $i_0$  z funkcją okna  $w$  opisane jest wzorem:

$$s_{i_0}^w = x(t)w(t_0 - t) \quad (2.6)$$

Do funkcji okna standardowo używa się funkcji Gaussa, Hamminga, Hanninga lub kwadratowej (choć tą ostatnią używa się tylko w wyjątkowych sytuacjach). Dokładniejszy ich opis znajduje się pod koniec tego rozdziału. Podczas analizy i syntezy wykorzystywane jest pojedyncze okno. Jak opisuje Manuel Davy w [28, s. 22–25] STFT w dziedzinie czasu i częstotliwości jest zdefiniowana na podstawie równania 2.6 i funkcji FFT jako transformata kolejnych okien:

$$STFT_x^W(t, f) = CFT_{s_{\frac{w}{t}}}(f) = \int_{-\infty}^{+\infty} x(\tau)w(t - \tau)e^{-j2\pi f\tau} d\tau. \quad (2.7)$$

Rozmiar okna dobierany jest zgodnie z charakterystyką analizowanych danych. Przykładem może być detekcja mowy, podczas którego okno powinno być o małej ilości próbek,

ponieważ dynamika tonacji przy każdym słowie jest bardzo wysoka. W celu odpowiedniego przeanalizowania wypowiedzi zsamplingowane fragmenty, które poddawane są STFT, powinny być o jak najmniejszej długości, aby analiza była wrażliwa na nawet najkrótsze zmiany w tonie, żeby właściwie wykryć formanty słów [27, 804–805].

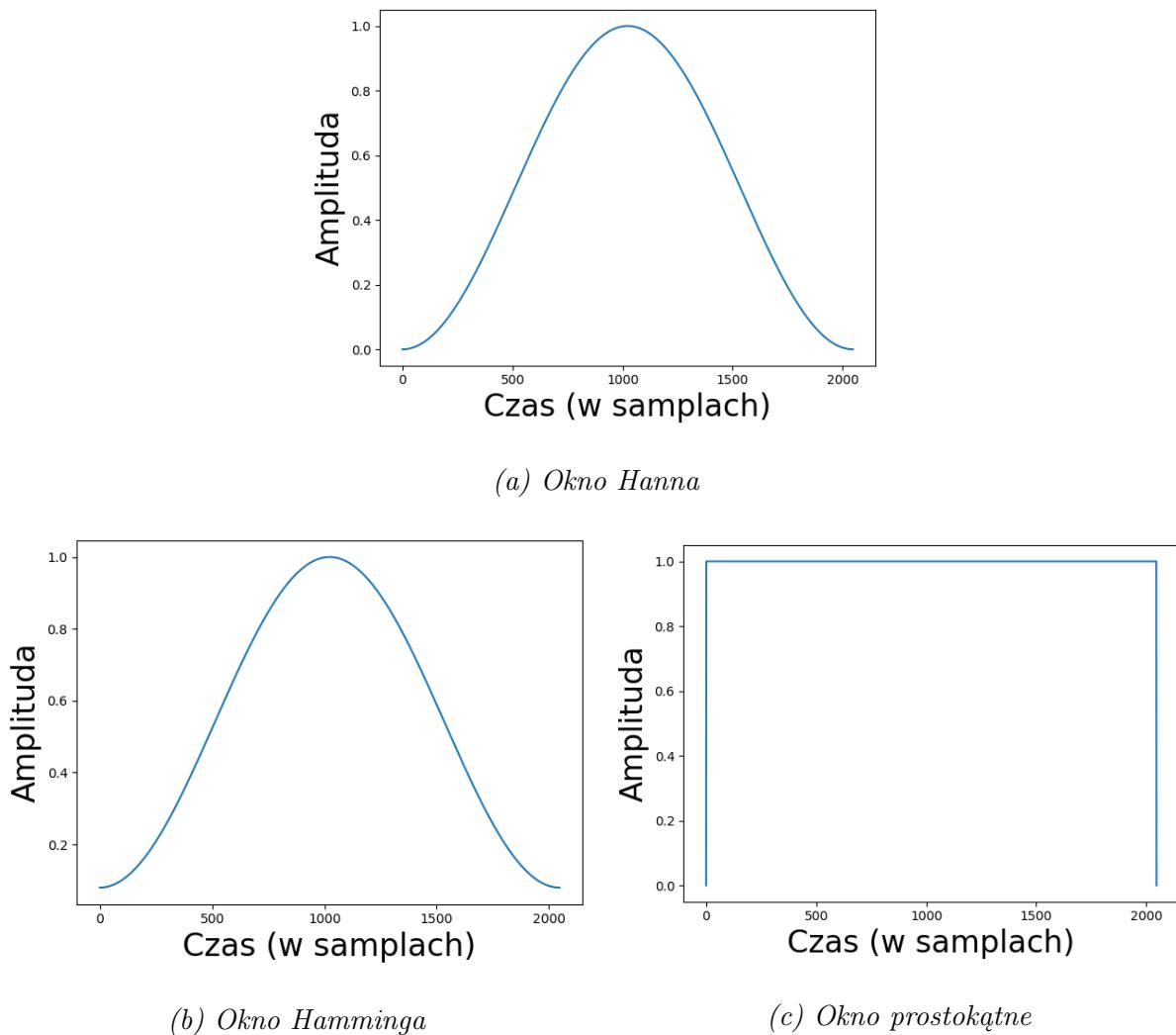
Równanie 2.7 nosi nazwę metody „przesuwającego się okna” MWM (z ang. *Moving Window Method*) w dziedzinie czasowej lub częstotliwości. STFT wykonywane w dziedzinie czasowej sprowadza się do wykonywania przekształcenia Fouriera na kolejnych fragmentach podzielonego sygnału poprzez przesuwanie okna  $\gamma(x)$ . W dziedzinie częstotliwości algorytm STFT wykonywany jest przy pomocy dwóch operacji:

1. Odwrotnym przekształceniem Fouriera fragmentu widma sygnału poprzez przesunięcie w domenie częstotliwości widma okna,
2. Przesunięciem w domenie częstotliwości sygnału czasowego otrzymanego z punktu 1 do częstotliwości zerowej poprzez wymnożenie go z  $\exp(-j2\pi ft)$ .

[54, s. 455–458].

Bardzo małe okna czasowe dają większą dokładność w domenie czasowej niż częstotliwości. Dla algorytmów wykrywających tonacje muzyki precyzja w częstotliwości jest dużo bardziej istotna niż precyzja w czasie. Technika, która pozytywnie wpływa na dokładność w obu tych płaszczyznach, zakłada nakładanie na siebie okna, przesuwając je mniej niż jego właściwą szerokość. W zależności od tego, jak mocno okno się na siebie nakłada, proporcjonalnie rośnie koszt obliczeniowy tego algorytmu (np. długość okna równa 2048 sampli, a przesunięcie równe 1024 sample spowoduje nakładanie się na siebie okien przy jednoczesnym podwojeniu ilości koniecznych obliczeń).

W momencie, gdy w algorytmie okno nakłada się na siebie, pewna porcja danych jest duplikowana. Ilość redundantnych danych jest proporcjonalna do szerokości nakładania się na siebie okien. Aby zminimalizować ten negatywny efekt, na okno nakładana jest funkcja dalej zwana funkcją okna. W ogólności, funkcja okna wydobywa odpowiednio przemnożone wartości wewnętrz jej zakresu (w oknie) i stałą wartość (zazwyczaj jest to zero) wszędzie poza obrębem zakresu okna. Dla okien czasowych można stosować wiele różnych funkcji matematycznych. W tabeli 2.1 przedstawione są funkcje  $w(t)$ , które często wykorzystywane są jako okna czasowe w analizie sygnału akustycznego, wraz z ich analitycznymi wzorami widm  $W(\omega)$ . Graficzną reprezentację okna prostokątnego, Hanny oraz Hamminga można zobaczyć na rysunku 2.2 [54, s. 87–90] [29].



Rysunek 2.2: Przykłady funkcji okna czasowego dla długości 2048 próbek.

Nazwa funkcji	$w(t)$	$W(\omega)$
Prostokątna	$p_T(t) = \begin{cases} 1 & \text{dla }  t  \leq T \\ 0 & \text{dla }  t  > T \end{cases}$	$2 \frac{\sin \omega T}{\omega}$
Trójkątne (Bartletta)	$q_T = \begin{cases} 1 -  t /T & \text{dla }  t  \leq T \\ 0 & \text{dla }  t  > T \end{cases}$	$T \left[ \frac{\sin(\omega T/2)}{\omega T/2} \right]^2$
Hanninga (Hanna)	$[0, 5 + 0, 5 \cos(\pi t/T)] p_T(t)$	$\frac{\pi^2 \sin(\omega T))}{\omega(\pi^2 - T^2 \omega^2)}$
Hamminga	$[0, 54 + 0, 46 \cos(\pi t/T)] p_T(t)$	$\frac{(1,08\pi^2 - 0,16T^2\omega^2)}{\omega(\pi^2 - T^2\omega^2))} \sin(\omega T)$

Tablica 2.1: Definicje funkcji okien czasowych  $w(t)$  i analityczne wzory ich widm  $W(\omega)$

### 2.2.4. Przetwarzanie transformacji Fouriera

Wynikiem FFT jest wektor liczb zespolonych, którego długość jest równa długości danych (suma długości okna i długość wypełnienia zerami). Urojona część wyniku jest niezbędna do odwrócenia transformaty, lecz nie jest niezbędna w procesie analizy sygnału. Większość algorytmów używających FFT przetwarza i skaluje wynik w celu pozbycia się niepotrzebnych informacji jednocześnie uwypuklając istotne cechy. Powszechnie używaną techniką podczas analizy sygnałów jest operowanie na logarytmie amplitudy spektrum [50, s. 501–507]. Polega to na zastosowaniu normy Gaussa na każdej z wartości spektrum, co w rezultacie zniweluje część urojoną komponentu, a następnie zastosowanie na wyniku logarytmu naturalnego. Logarytm użyty jest do normalizacji i wyrównania danych przy zachowaniu relatywnych wartości. Wzór na logarytm amplitudy spektrum można przedstawić następująco:

$$S_k^w(t, f) = \ln \|STFT_k^w(t, f)\| \quad (2.8)$$

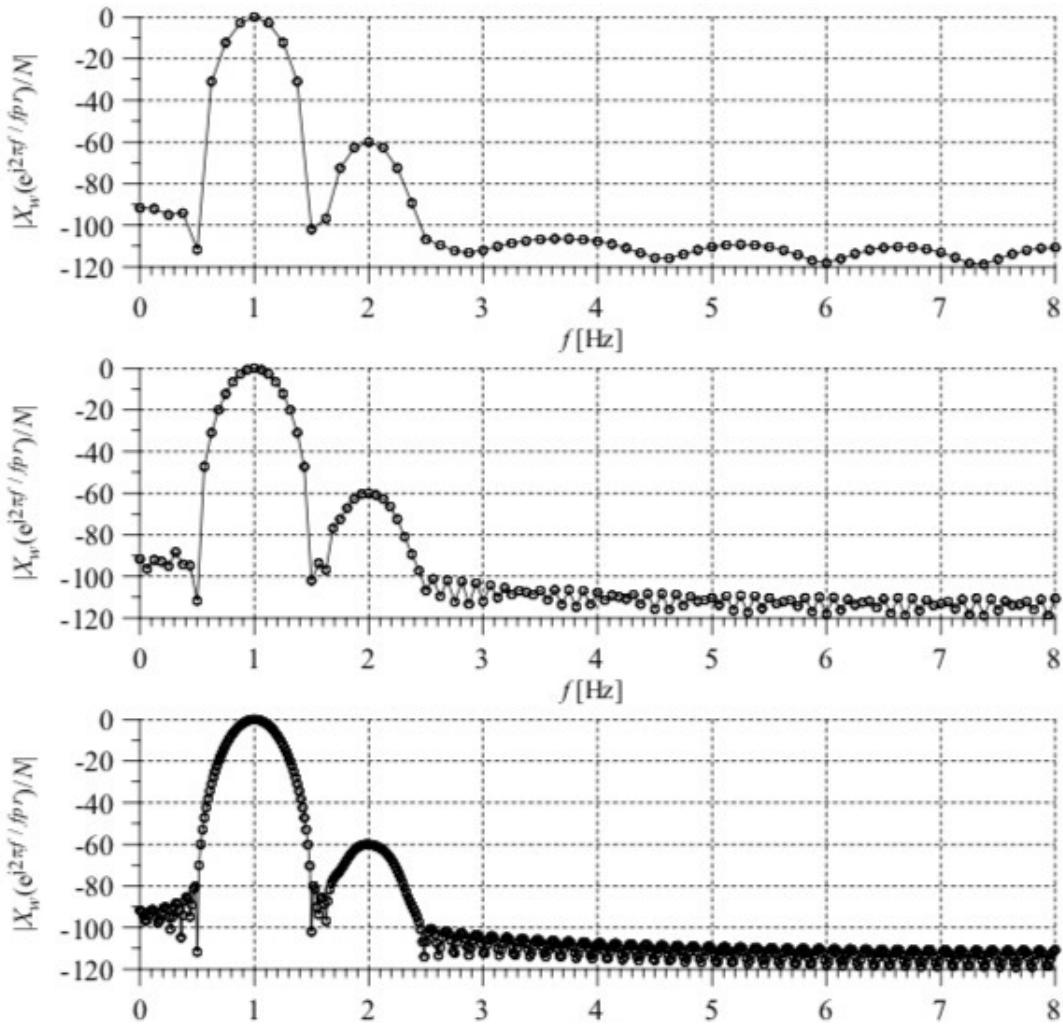
gdzie  $STFT_k$  jest  $k$ -tym komponentem spektrogramu. Możliwe jest również użycie potęgowania zamiast logarytmu, co zostało dokładniej opisane w sekcji 3.2.

Przy analizie częstotliwości składowych widmowych sygnału na uwadzę należy mieć twierdzenie Nyquista, znane też jako twierdzenie Kotielnikowa - Shannona lub *twierdzeniem o próbkowaniu*. Mówiąc o istnieniu częstotliwości Nyquista, która opisana jest wzorem:

$$f_{Nyquist} = \frac{1}{2}v \quad (2.9)$$

gdzie  $v$  jest częstotliwością próbkowania. Częstotliwość ta jest maksymalną częstotliwością, jaką można wydobyć z sygnału o danym  $v$ . Odwracając proces, wzór 2.9 mówi o tym, że częstotliwość próbkowania  $v = \frac{1}{\delta t}$  musi być dobrana tak, aby była dwa razy większa od maksymalnej częstotliwości występującej w sygnale.

Przy modelowaniu systemu transkrypcji często trzeba podjąć trudną decyzję, czy zwiększyć złożoność obliczeniową i osiągnąć lepszą aproksymację widma podczas FFT poprzez zwiększenie częstotliwości próbkowania lub zastosowanie interpolacji, czy zmniejszyć dokładność w ceku otrzymania mniejszej złożoności obliczeniowej. Istnieje jednak metoda, która prowadzi do kompromisu tych dwóch podejść. Polega ona na uzupełnieniu na końcu zerami analizowanego fragmentu sygnału po nałożeniu funkcji okna. Na tak rozszerzonych danych STFT wykonywane jest analogicznie. Podejście to daje w wyniku gęstsze próbko-



Rysunek 2.3: Przykład analizy częstotliwości z wykorzystaniem STFT. Każdy z wykresów przedstawia to samo okno po uzupełnieniu na końcu zerami do długości  $N_{FFT} = 128, 256, 1024$  (kolejno od góry). Wykres z [54, s. 224].

wanie widma przy mniejszej złożoności obliczeniowej. Zwiększoną poprzez konkatenację zer rozdzielcość widma można zaobserwować na rysunku 2.3 [54, s. 221–224].

### **3. Estymacja częstotliwości fundamentalnej w sygnale monofonicznym**

W tym rozdziale opisane są metody estymacji składowej fundamentalnej w sygnale akustycznym. Jest to jeden z kroków wymaganych do pełnej transkrypcji utworu muzycznego, a przez wielu badaczy analizy akustycznej uważany za najważniejszy [2, s. 408–415]. Wysokość dźwięku jest niezwykle istotną informacją z punktu widzenia muzyka, a sposób zagrania poszczególnych nut determinowany jest właśnie przez F0. Nawiązując do informacji opisanych w rozdziale 1 aby poprawnie wykryć F0 należy przeanalizować szereg czynników związanych z sygnałem akustycznym.

Samo rozpoznanie wysokości tonu jest kategoryzowane w zależności od specyfiki sygnału wejściowego na transkrypcję utworów monofonicznych (dla sygnału bez równoległych dźwięków) oraz transkrypcję utworów polifonicznych (dla sygnałów z równoległymi dźwiękami). Te drugie są często dalej dzielone na transkrypcję sygnałów mono- i multiinstrumentalnych (w zależności od tego, czy w sygnale występuje tylko jeden lub więcej instrumentów o różnych barwach dźwięku).

Dalsze rozważania będą przebiegać z założeniem, że przetwarzane sygnały są z podzbioru muzyki o systemie równomiernie temperowanym, lecz bez wyszczególnionego gatunku muzycznego. Nie będziemy również skupiali się na wykryciu tak zwanych dźwięków o niezależnej wysokości, do których zaliczają się elementy perkusyjne. Ważnym założeniem, które jest stosowane w tym rozdziale, jest analiza jedynie sygnałów monofonicznych. W odróżnieniu do krzywych reprezentujących muzykę polifoniczną, w analizowanych utworach będzie grany tylko jeden dźwięk na pojedyńczym instrumencie, bez równoległych dźwięków. Algorytmy analizujące muzykę wielotonową są opisane w rozdziale 4. Jest to bardzo duże uproszczenie, ponieważ konkurujące sygnały dźwiękowe nakładają się na siebie, zmuszając algorytm analizujący do rozróżnienia i odseparowania ich przed faktyczną analizą F0.

#### **3.1. Funkcja autokorelacji**

Domena czasu jest najbardziej naturalną postacią sygnału dźwiękowego, ponieważ w takiej właśnie postaci jest odbierane przez ludzkie ucho - jako ciśnienie akustyczne zmienne w czasie. Wiele z pierwszych badań, jakie prowadzono w kontekście transkrypcji sygnału fonicznego, wykorzystywało tą postać sygnału, między innymi w przytoczonej

wcześniej pracy [13]. Algorytm ten polegał na znajdywaniu wzorców w krzywej sygnału. Regularności, które powtarzały się okresowo co  $T$ , były analizowane jako kandydaci F0. Analiza sygnału w domenie czasu i wyliczenia na nim nie są częstą praktyką, ale podejście do znajdywania wzorców i regularności jest powszechna we współczesnych algorytmach. Najbardziej dosłowne odwzorowanie tej idei można znaleźć w metodologiach opartych na korelacji, których celem jest znalezienie wysokości dźwięku [42, s. 41–44].

Autokorelacja jest narzędziem matematycznym stosowanym w przetwarzaniu sygnałów. Służy ono do analizy serii wartości. Ta statystyczna miara odwzorowuje podobieństwo pomiędzy kolejnymi wartościami tej samej zmiennej. Funkcja ta ma za zadanie wykrycie regularności w analizowanych danych [54, s. 14–17]. Proces autokorelacji polega na badaniu korelacji wejściowych danych z tymi samymi danymi, ale przesuniętymi o pewną wartość (dalej określone jako przesunięcie, z ang. *lag*). Duży współczynnik korelacji o odpowiednim przesunięciu może świadczyć o powtarzającym się wzorze w sygnale o częstotliwości danego przesunięcia.

W pracy naukowej [43] Lawrence używa funkcji autokorelacji do wykrycia częstotliwości fundamentalnych w zadany sygnale monofonicznym. Zakłada on, że badany sygnał jest quasi-periodyczny, co oznacza, że sygnał posiada cechy pojawiające się w prawie równych odstępach, lecz nie musi być on dokładnie periodyczny. Sygnał w pełni periodyczny jest przedmiotem analizy czysto teoretycznej, ponieważ sygnał akustyczny musiałby pozostać bez zmian w amplitudzie i częstotliwości przez cały okres, podczas gdy sygnał quasi-periodyczny zakłada zmiany w strukturze sygnału względem czasu. Funkcja autokorelacji na takim sygnale byłaby funkcją długości okresu opisującą zgodność przesuniętego o ten okres sygnału z sygnałem bazowym.

Wzór na ogólną postać funkcji autokorelacji od długości okresu  $h(t)$  można zapisać jako:

$$R(k) = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N h(n)h(n+k) \quad (3.1)$$

gdzie  $k$  jest opóźnieniem,  $n$  reprezentuje moment w czasie a  $N$  reprezentuje długość analizowanych danych. Jak można zaobserwować we wzorze 3.1 założeniem jest, aby sygnał bazowy był nieskończoność długi (tak samo jak przy TF opisanej w sekcji 2.2). Do rozważań w kontekście analizy sygnału fonicznego będziemy używać wersji krótkoczasowej tego wzoru, operującej na oknach danych (analogicznie do STFT opisanej w sekcji 2.2.3),

która została opisana w [50, s. 503] jako:

$$R(i, k) = \sum_{j=m}^{m+N-k-1} h_j h_{j+k}, m = iz \quad (3.2)$$

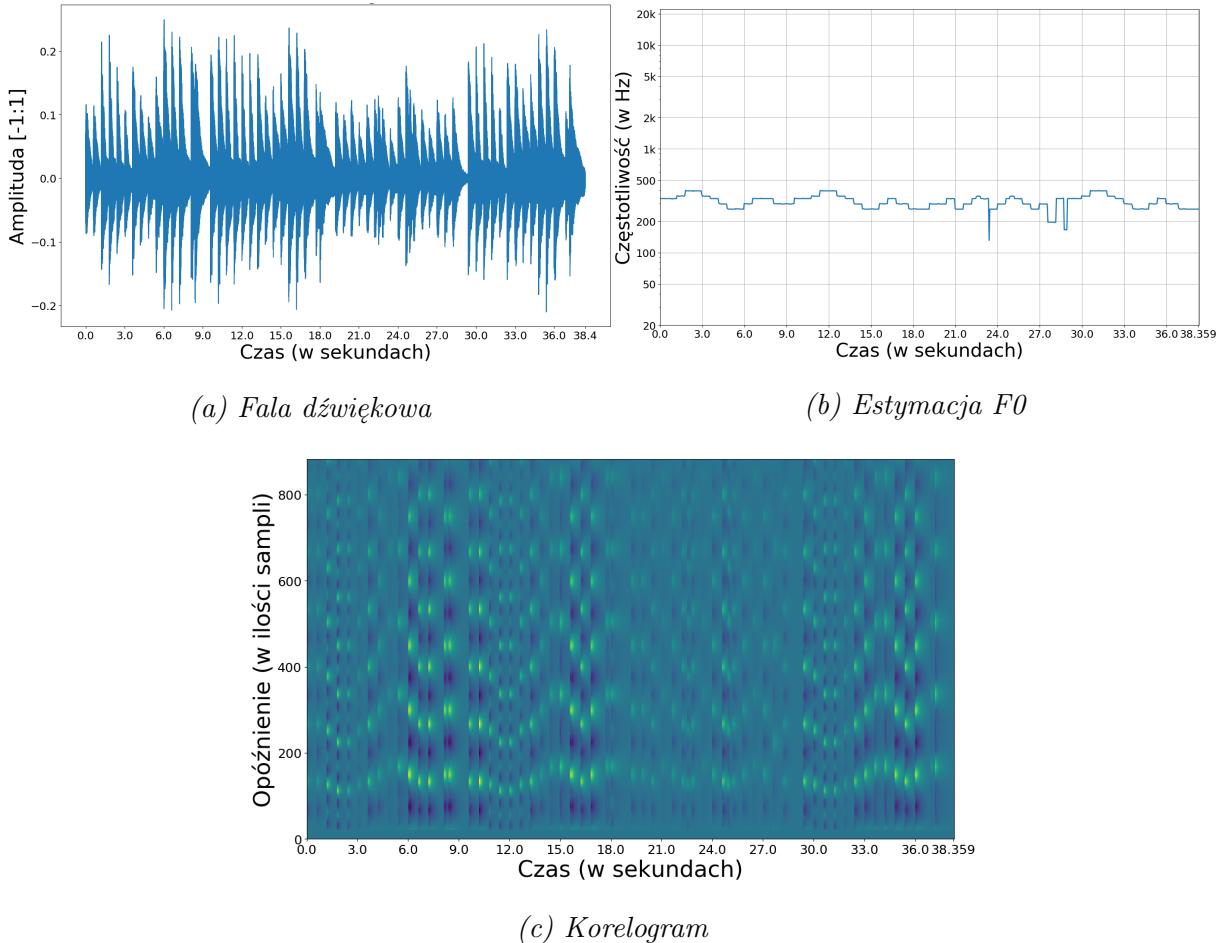
gdzie  $i$  jest indeksem badanego okna,  $z$  jest odstępem pomiędzy oknami audio (w ilości sampli), a  $N$  jest długością okna. Zgodnie z twierdzeniem Nyquista opisanym w sekcji 2.2.3 długość okna powinna być co najmniej dwa razy większa niż największe oczekiwane opóźnienie, jakie może wystąpić w analizowanym sygnale akustycznym. Implementacja estymacji F0 przy pomocy autokorelacji została opisana w sekcji 5.1.

Z jednego okna danych w wyniku wyznaczane jest przesunięcie, które ma największy współczynnik korelacji. Dlatego, że domena czasu w dyskretnym sygnale na jakiej operuje autokorelacja jest tak naprawdę domeną ilości sampli, aby uzyskać wynik w postaci częstotliwości należy podzielić częstotliwość samplowania przez wyznaczone przesunięcie:

$$fq = \Delta / lag \quad (3.3)$$

Podstawowym założeniem algorytmów wykrywania F0 z wykorzystaniem autokorelacji jest to, że analizowany sygnał jest monofoniczny. Jest to spowodowane tym, że periodyczność, jaką cechuje się pojedyńcza krzywa quasi-periodyczna, jest modulowana albo nawet całkowicie neutralizowana po nałożeniu na nią innej krzywej. Założymy, że został nagrany sygnał z dwoma nutami o różnych wysokościach grannymi równolegle w tym samym czasie. Okresami, w których te sygnały w izolacji byłyby powtarzalne to  $T_1$  i  $T_2$ . Po nałożeniu się na siebie, wynikowy sygnał nie byłby periodyczny w żadnym z tych interwałów, chyba że jeden byłby wielokrotnością drugiego. Jest to opisane dokładniej w sekcji 4.1.

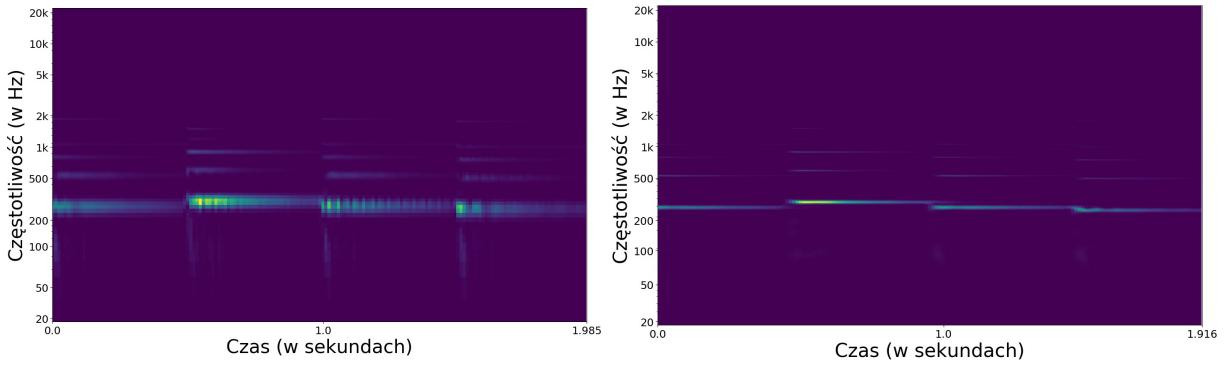
Wyniki estymacji F0 mają tendencje do bycia wielokrotnościami faktycznego F0 sygnału, co można zaobserwować na wykresie 3.1b. Dzieje się tak, ponieważ wysoki współczynnik korelacji, jaki uzyska krzywa o periodyczności równej  $T$  dla opóźnienia  $T$  będzie relatywnie podobny do wyniku autokorelacji tego sygnału dla opóźnienia  $2T$ ,  $3T$  itd. Jest to spowodowane tym, że wzorzec występujący co  $T$  występuje w szczególności co  $2T$ ,  $3T$  itd. [28, s. 231–244] Można to zaobserwować w koreogramie przedstawionym na rysunku 3.1c.



Rysunek 3.1: Wynik autokorelacji na kompozycji „Oda do radości” IX symfonii Beetho-vena zagranej w sposób monofoniczny na pianinie. Użyto okna Hanna o długości 2048 sampli z odstępami długości 2048 sampli. Wybór F0 polegał na wybraniu największego współczynnika korelacji w oknie. Jaśniejsze obszary na koreogramie oznaczają większą magnitudę.

## 3.2. Analiza TFR

Analiza sygnału w reprezentacji średniego poziomu TFR, w przeciwieństwie do operowania na nieprzetworzonym sygnale cyfrowym, jak miało to miejsce w sekcji 3.1, skupia się na badaniu relacji w domenie częstotliwości danego sygnału. Reprezentacja TRF może SFTF na kolejnych fragmentach (oknach) sygnału. Docelową postacią jest zbiór okien tworzących spektrogram, ale w celu jego uzyskania należy SFTF przedstawić



(a) Spektrogram z długością okna 1024 sample (b) Spektrogram z długością okna 4096 sample

Rysunek 3.2: Spektrogramy sygnału sekwencji C3-D3-C3-B2 zagranej na pianinie wyliczone przy pomocy STFT z oknem Hann'a długości odpowiednio 1024 i 4096 dla wykresów (a) i (b) z odstępami o długości 512 sampli. Jaśniejsze obszary oznaczają większą magnitudę.

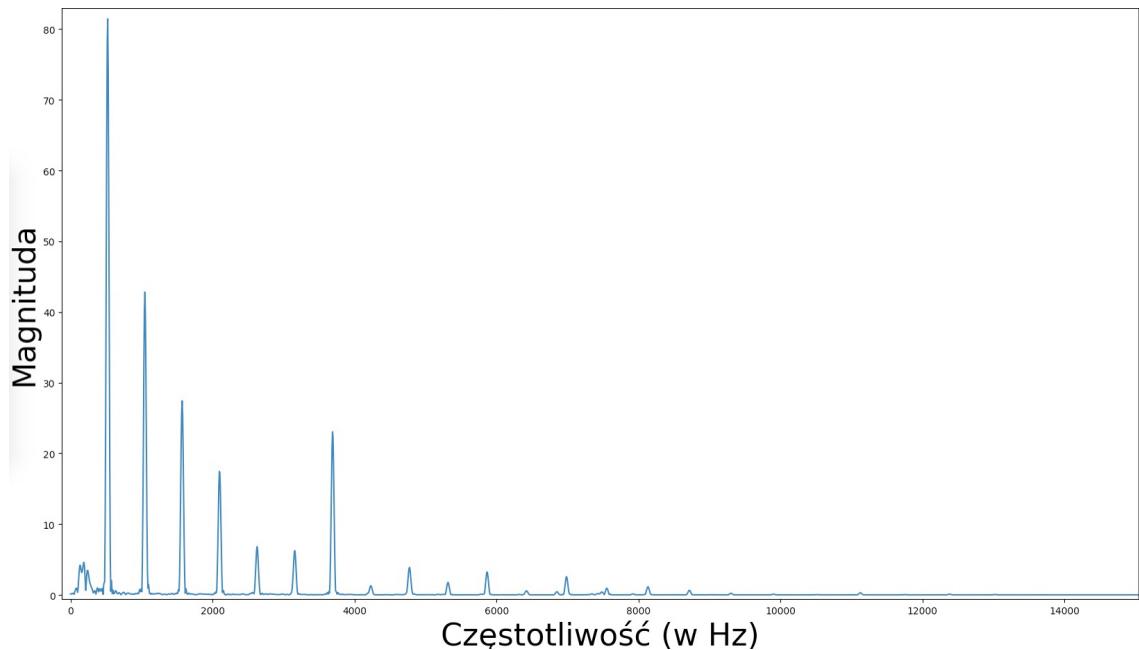
jako wektor energii. Jednym ze sposobów, w jaki można uzyskać taką postać, jest to wykonanie kwadratu modułu z STFT:

$$SP_x^w(t, f) = |STFT_x^w(t, f)|^2 \quad (3.4)$$

Wynik powyższego równania 3.4 nazywany jest potocznie widmem mocy (z ang. *power spectrum*). W celu uzyskania spektrogramu używa się również innych metod, jak na przykład logarytmu z modułu (co zostało opisane w sekcji 2.2.4). Wybór metodyki zależy od tego, które cechy sygnału wejściowego są bardziej, a które mniej istotne w kontekście algorytmu analizującego. Użycie widma mocy wiąże się z uwydatnieniem już wyraźnych cech, wraz z zwiększeniem istotności szumu w sygnale.

Zmiana funkcji okna  $w$  zmieni specyfikę STFT, co przekłada się na inny wynikowy spektrogram. Długość użytego okna jest także istotną cechą spektrogramu. Zmiany w dokładności poszczególnych wartości spektrogramu przy różnych długościach okna opisane są w sekcji 2.2.3 i przedstawione są na rysunku 3.2.

Podczas analizy spektrogramu, naiwnym podejściem byłoby wybranie za F0 ten pik w spektrum, który ma największą energię. Niestety jednak nie jest to zawsze prawidłowy wybór - częstotliwość fundamentalna wcale nie musi być tą, która ma najczęściej energii w sygnale akustycznym. Ze względu na charakterystyki różnych instrumentów,



Rysunek 3.3: Przykład pojedyńczego komponentu spektralnego ze spektrogramu z rysunku 3.2 o długości okna 4096.

formanty, szum i inne zniekształcenia opisane dokładniej w sekcji 1.4 kolejne harmoniczne mogą być wzmacnione ponad F0.

Analiza spektrogramu w celu znalezienia F0 polega na znalezieniu korelacji w rozmieszczeniu harmonicznych względem siebie. Zgodnie z regułą harmoniczności, harmoniczne sygnału mają częstotliwości będące wielokrotnościami F0. Są one równomiernie rozmieszczone na przestrzeni całego spektrum, co powoduje, że pojedyncza nuta jest sama w sobie periodyczna w domenie częstotliwości, co można zauważyć na rysunku 3.3. Ta charakterystyka jest powszechnie używana w algorytmach wykrywających częstotliwości fundamentalne w sygnałach akustycznych.

### 3.3. Analiza cepstralna

Z uwagi na fakt, że spektrum samo w sobie jest periodyczne, możliwe jest użycie metodologii wykrywającej cykliczności na spektrum. Taki pomysł mieli już w 1963 Bogert, Healy i Tukey w [4]. Wynaleźli oni narzędzie matematyczne przeznaczone dla danych zawierających echa lub pogłosy fundamentalnej częstotliwości, która nie jest znana. Cepstrum mocy używane jest do wyznaczenia czasu początku danego dźwięku, jego echa

i powiązanych amplitud. Analiza na zespółonym cepstrum pozwala na powrót do pierwotnego kształtu badanego sygnału [6] [42, s. 46–50].

W roku 1967 cepstrum zostało zastosowane do wykrywania wysokości dźwięku w ramach pracy [35]. Jak opisuje Oppenheim i Schafer w pracy naukowej [38, s. 95–99], która dotyczy historii tego narzędzia, równolegle i niezależnie z tymi badaniami powstała praca doktorska [37] badająca klasę nielinearnych technik przetwarzania sygnałów poprzez koncepcje homomorficznego mapowania pomiędzy grupami algebraicznymi i przestrzeniami wektorowymi. Ma ona szczególne zastosowanie w metodologii cepstralnej, pokrywając się wnioskami i przyspieszając tempo jego rozwoju. Dalsze prace Oppenheima opisują między innymi charakterystyki splotu harmonicznych, które są bardzo podobne do spektrum logarytmu spektrum. Odkrycia te przyczyniły się do, między innymi, stosowania cepstrum na oknach czasowych (tak jak STFT) czy opisanie zespółonego cepstrum, które pozwala na odwrócenie tej transformacji, otwierając drogę do takich metod jak filtrowanie w domenie cepstralnej.

Oryginalnie analiza cepstralna, która opisana jest w przytoczonej powyżej pracy [4] została zdefiniowana jako „spektrum mocy z logarytmu spektrum mocy”. Nawiązując do wzoru 2.8, cepstrum mocy okna o indeksie  $k$  z oknem  $w$  liczy się jako:

$$C_k^w(t, f) = |(S_k^w(t, f)|^2. \quad (3.5)$$

Pierwotnym zastosowaniem tego narzędzia było wykrycie echa w sygnale sejsmicznym. Podejście to było bardziej odporne na barwę sygnału od wcześniej używanej funkcji auto-korelacji (opisanej w sekcji 3.1). Zastosowanie to było czysto analityczne i nie wymagało od algorytmu możliwości przywrócenia analizowanych danych do domeny czasu. Mimo, że jednym z autorów pierwotnej pracy o cepstrum jest Tukey, który także pracował nad FFT, cepstrum zostało opublikowane na dwa lata przed pracą opisującą algorytm FFT [8], przez co jego potencjał nie był jeszcze znany. Choć w pierwotnej pracy nad cepstrum zdefiniowane było pojęcie filtrowania w domenie cepstralnej pod nazwą „lifter”, było zastosowane później z użyciem filtru konwolucyjnego [44, s. 1–2].

Nazwa cepstrum powstała poprzez odwrócenie pierwszych liter wyrazu *spectrum* (z ang. spektrum). Szereg terminologii został opisany na potrzeby zdefiniowania narzędzia cepstrum w pierwotnej pracy [4], które do dziś są używane w celu wyróżnienia domen cepstralnych:

---

<i>rahmonics</i>	zamiast	<i>harmonics</i> (z ang. harmoniczność)
<i>liftering</i>	zamiast	<i>filtering</i> (z ang. filtrowanie)
<i>quefrency</i>	zamiast	<i>frequency</i> (z ang. częstotliwość)
<i>gamnitude</i>	zamiast	<i>magnitude</i> (z ang. wielkość)

Wy tłumaczeniem tej lingwistyki, w której znajome wyrazy były modyfikowane w celu uzyskania czegoś podobnego, ale będącego nowymi słowami, jest odróżnienie tych pojęć od standardowych definicji. Jak wyjaśnione jest to w pracy na temat historii cepstrum:

„Ogólnie rzecz biorąc, działamy po stronie częstotliwości w sposób powszechnie stosowany po stronie czasu i odwrotnie” [38, s. 95] (tłumaczanie własne)

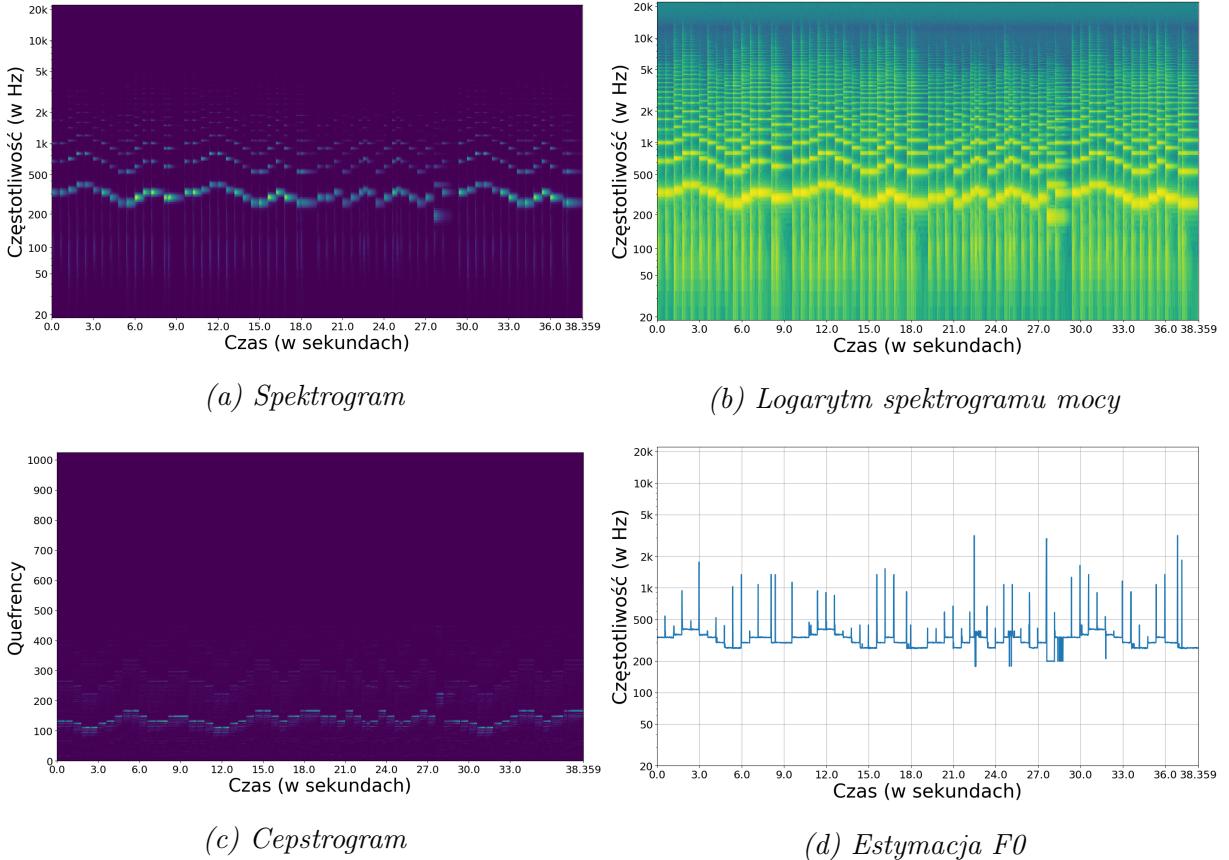
Cepstrum posiada pik tam, gdzie oryginalny sygnał w przestrzeni czasu - amplitudy posiada echo. Ta nowa reprezentacja spektralnej domeny nie jest już domeną częstotliwości, ani też nie jest to domena czasu. Przez ten fakt, aby nie mylić tych pojęć z pozostałymi Bogert nazwał domenę cepstrum jako domena *quefrency*, która tak naprawdę jest domeną częstotliwości spektralnych pików oryginalnego sygnału [44, s. 1–4].

W przypadku lifterowania, czyli filtrowania w domenie quefrencji, używany jest wzór cepstrum sygnału ciągłego  $x(t)$ :

$$Cep_x(\tau) \text{ ICFT}_{\log(|X|)}(\tau) = \int_{-\infty}^{\infty} \log(|X(f)|) e^{j2\pi f \tau} df, \quad (3.6)$$

gdzie  $X(f)$  oznacza TF. Zgodnie z tym wzorem filtrowanie sygnału w domenie częstotliwości przez wynik  $X(f)H(F)$  staje się, po wzięciu  $\log$ , sumą  $\log[X(f)] + \log[H(f)]$ , gdzie  $H(f)$  jest filtrem częstotliwościowym [28, s. 25–27].

Postać cepstralną sygnału akustycznego można bezpośrednio wykorzystać do znalezienia częstotliwości fundamentalnej w monofonicznym sygnale. Podejście to operuje na cepstrum mocy, które opisane jest wzorem 3.5 dla kolejnych okien czasowych. Tak pozyskane cepstrum badane jest w kontekście kandydatów F0 i wybierany jest ten, o największej magnitudzie. Dokładna implementacja opisana jest w sekcji 5.2. Metoda ta jest podatna na wyznaczanie wielokrotności F0, zamiast faktycznej częstotliwości fundamentalnej, tak jak metoda autokorelacji 3.1. Ten defekt można zaobserwować na ilustracji 3.4. To podejście nie radzi sobie też z sygnałami zawierającymi dużą ilość szumów, co jest przedstawione w [29, s. 233–235]. Mimo tych wad, metodyka cepstrum jest powszechnie

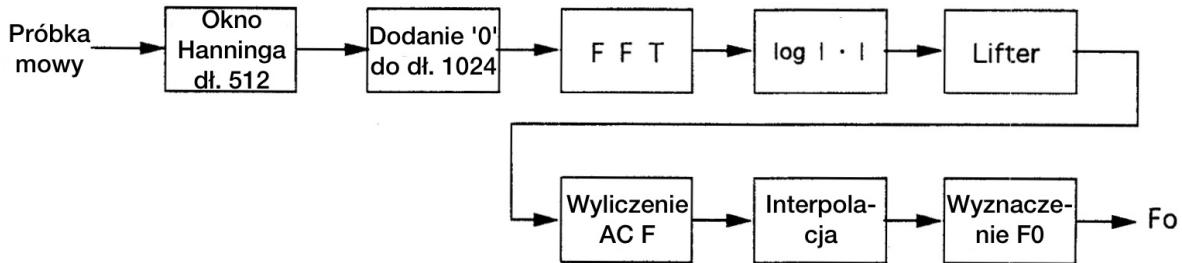


Rysunek 3.4: Wynik cepstrum na na „Odzie do radości” IX symfonii Beethovena zagranej w sposób monofoniczny na pianinie. Użyto okno Hanna o długości 2048 sampli z odstępami długości 512 sampli. Wybór F0 polegał na wybraniu największego współczynnika cepstralnego w oknie. Jaśniejsze obszary w spektrogramach i cepstrogramie oznaczają większą magnitudę.

stosowana w analizie sygnałów. Jej ważnym zastosowaniem jest filtrowanie przy pomocy lifteringu, które używane jest między innymi w opisany w sekcji 3.4 algorytmie ACLOS.

### 3.4. ACLOS

Do badanie cykliczności spektrum można wykorzystać narzędzia autokorelacji. Omówiona wcześniej metoda badająca cykliczność fali akustycznej poprzez wyliczanie autokorelacji (3.1) użyta w domenie częstotliwości została opisana w 1996 roku w pracy [29] pod nazwą **ACLOS**. Oba te algorytmy polegają na zasadzie cykliczności fal akustycznych i harmonicznych F0, próbując wykryć dokładny odstęp pomiędzy cyklami.

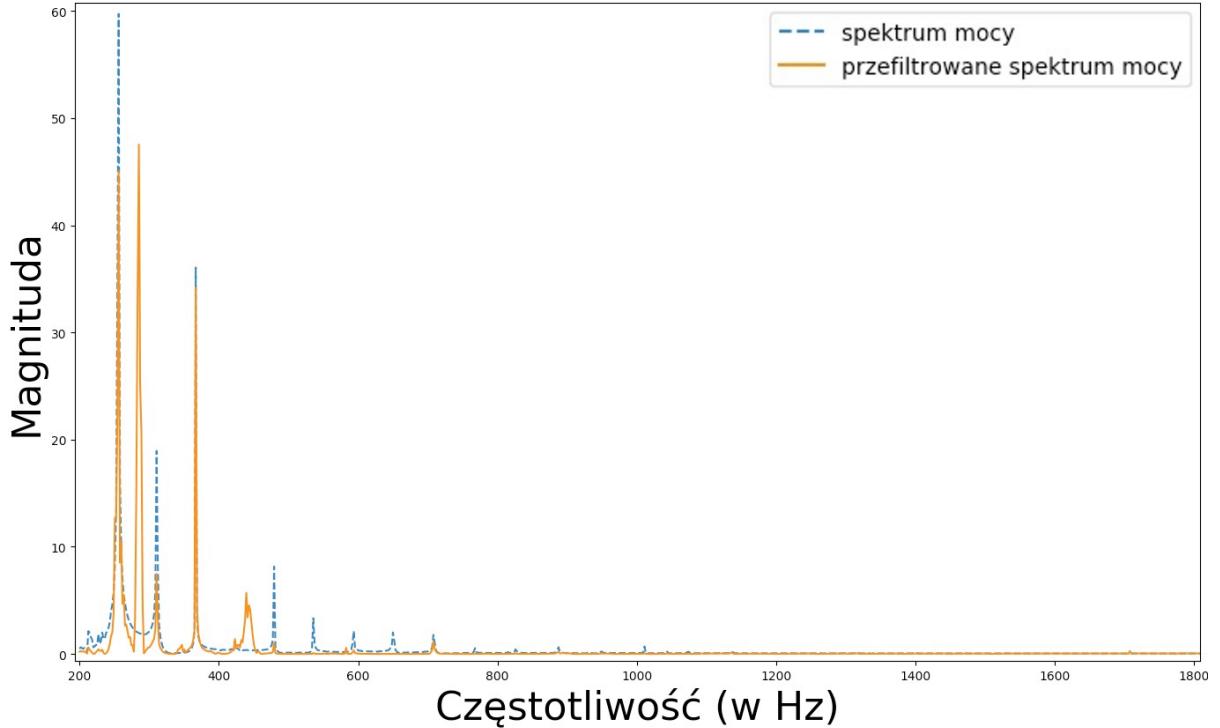


Rysunek 3.5: Schemat działania algorytmu ACLOS do wykrywania wysokości dźwięku.  
Schemat z [29, s. 233], tłumaczenie własne.

Nazwa algorytmu ACLOS jest skrótem od *AutoCorrelation of LOg Spectrum*, co oznacza autokorelację na logarytmie widma (tłumaczenie własne). Motywacją do stworzenia ACLOS było odkrycie takiego algorytmu, który wykorzysta zalety cepstrum i jednocześnie będzie bardziej odporne na szумy w badanym sygnale. Dodatkowo, autorzy ACLOS przytaczają prace naukową [30], która również wykorzystuje autokorelacje na widmie sygnału akustycznego, z tą różnicą, że nie stosuje się na nim logarytmowania a proces autokorelacji determinowany jest procedurą decyzyjną zależną od pików na poszczególnych pasmach sygnału, co czyni ten algorytm dużo bardziej złożony niż ACLOS.

Schemat działania algorytmu ACLOS dla okna Hanninga o długości 512 sampli przedstawiony jest na rysunku 3.5. W zależności od długości okna zmieniać się będzie wynikowa częstotliwość i maksymalna częstotliwość, jaką można wykryć w sygnale (zgodnie z równaniem Nyquista przedstawionym w 2.9). Do okna dodana jest odpowiednia ilość zer jako margines, które mają na celu zwiększenie częstotliwości widma (jak zostało opisane w sekcji 2.2.4). W oryginalnej pracy margines ten ma długość 512, aby wynikowe okno miało długość równą 1024. Na rozszerzonym o wektor zer oknie wyliczana zostaje szybka transformata fouriera, a z niej wynik przetwarzany jest do logarytmu spektrum mocy zgodnie ze wzorem 3.4. Następną operacją, jaka jest wykonywana na analizowanych danych, jest filtrowanie w domenie cepstrum. Liftering ma tu na celu wyrównanie spektrum i zmniejszenie wpływu, jaki ma na wynik szum. Wynik działania lifteringu można zaobserwować na rysunku 3.6. Na tak przygotowanym oknie wykonywana jest funkcja autokorelacji  $R(j)$ , która ma postać:

$$R(j) = \frac{1}{N} \sum_{i=0}^{N-1} f(i)f(i+j) \quad j = 0, 1, \dots, M \quad (3.7)$$

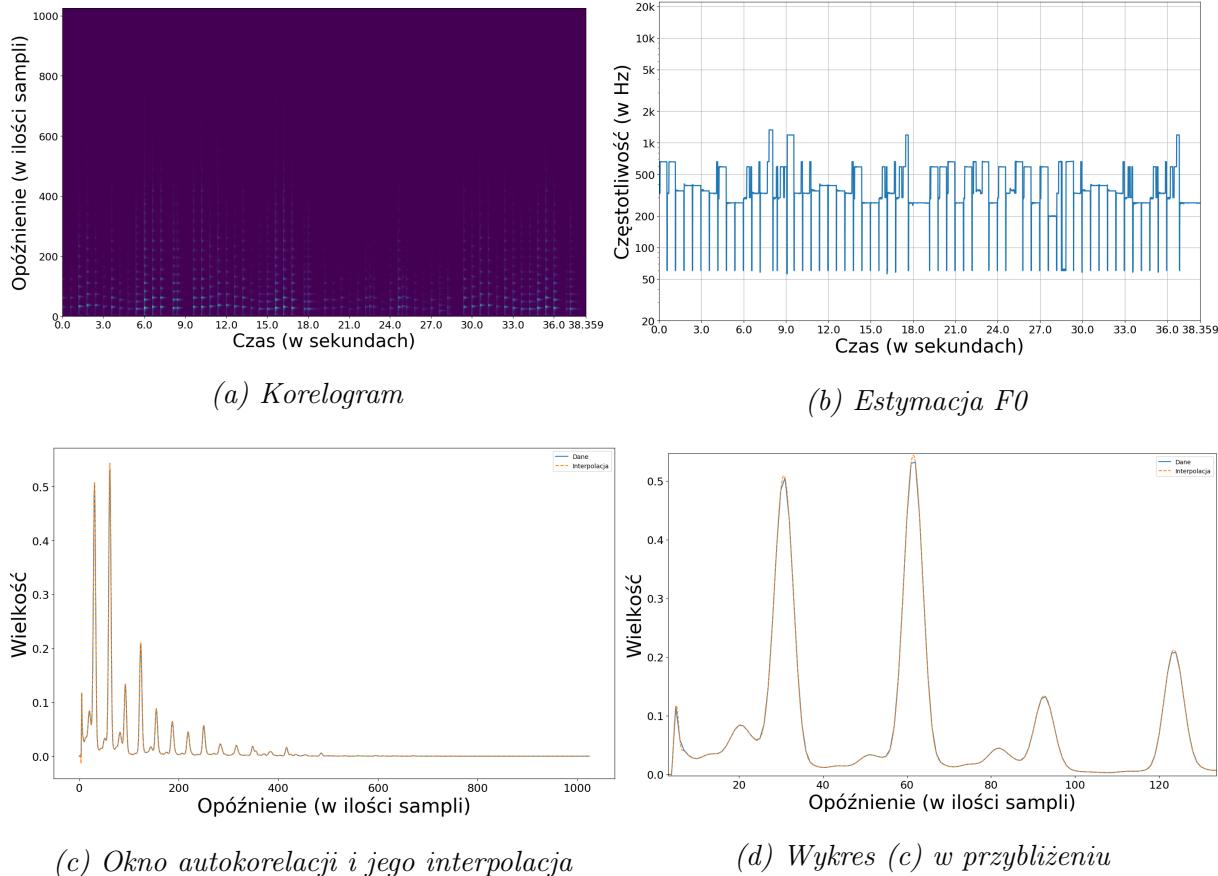


Rysunek 3.6: Wykres przedstawiający spektrum na pojedyńczym oknie przed i po nałożeniu lifterowania ze współczynnikiem równym 10

gdzie  $f(i)$  jest i-tym komponentem lini widmowej,  $N$  oznacza zakres harmonicznych, który brany jest pod uwagę,  $j$  jest częstotliwością dla której badany jest współczynnik autokorelacji a  $M$  oznacza maksymalną częstotliwość, jaką może posiadać F0 w analizowanym sygnale [42, s. 50–51].

Algorytm ACLOS po wyliczeniu wektora autokorelacji dla danego okna zakłada wyznaczenie F0 na podstawie maksymalnego wyniku z równania 3.7. Szerokość pasma pojedyńczego spektralnego komponentu zależna jest od przyjętej długości okna, a pasmo to jest zakresem, w którym znajduje się znalezione F0. Autorzy używają interpolacji wokół wartości wyznaczonej przez autokorelacje, co w rezultacie umożliwia uzyskanie większej dokładności wyniku.

Wyniki ACLOS w wyznaczaniu F0 są podobne do tych, wyznaczonych przez metodę Cepstrum (3.3) co można zaobserwować na rysunku 3.7. Większa niedokładność na początku kolejnych wysokości jest spowodowana charakterystyką użytego pianina, którego barwa podczas narastania amplitudy dźwięku jest nieliniowa. ACLOS jest uważane za metodę bardziej odporną na szum w analizowanym sygnale niż metoda Cepstrum [29].



Rysunek 3.7: Wynik ACLOS użytego na kompozycji „Oda do radości” IX symfonii Beetho-vena zagranej w sposób monofoniczny na pianinie. Użyto okna Hanna o długości 2048 sampli z odstępami długości 512 sampli. Wykresy (c) i (d) przedstawiają wynik równania 3.7 na tym samym oknie po zastosowaniu lifteringu. Jaśniejsze obszary w cepstrogramie oznaczają większą magnitudę.

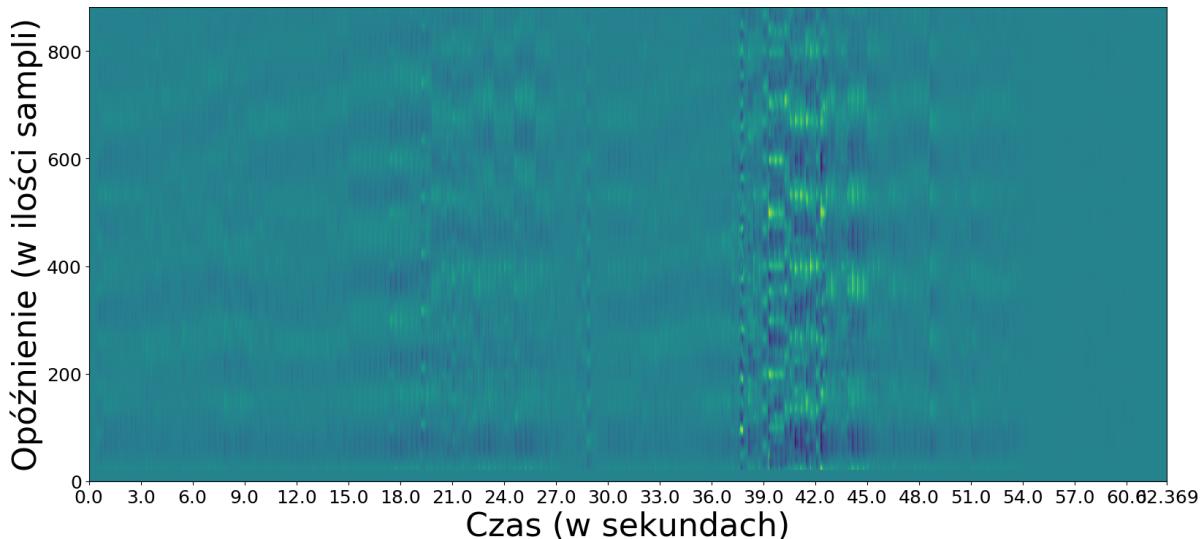
## **4. Estymacja wielu częstotliwości fundamentalnych w sygnale polifonicznym**

W tym rozdziale opisane są metody estymacji składowej fundamentalnej w polifonicznym sygnale akustycznym. Sygnał polifoniczny to taki, w którym występują co najmniej dwa równoległe dźwięki w tym samym czasie. Problem ten jest znacznie bardziej złożony od wyznaczania pojedyńczego F0 w sygnale monofonicznym opisanego w 3. Wiele z własności sygnałów akustycznych, jak periodyczność czy rozróżnialna barwa, nie są niezawodnym źródłem informacji w celach transkrypcji wielotonowej, ponieważ nakładające się na siebie dźwięki modulują te cechy dźwięków zniekształcając ich pierwotną formę.

Współczesna muzyka zachodnia jest w dużym stopniu ustrukturyzowana zarówno w domenie czasu jak i częstotliwości. Tempo i rytm, w którym dany utwór jest grany, są informacjami zawartymi w domenie czasu. Dzięki wzorom, które są zachowywane w muzyce, można te informacje wywnioskować. W domenie częstotliwości natomiast widoczne są zależności pomiędzy poszczególnymi wysokościami dźwięków. Pojedyńczej zagranej nutie towarzyszy, poza częstotliwością fundamentalną, szereg wydźwięków, co było już wielokrotnie udowadniane, jak na przykład w [33, s. 1325–1326]. Vibracje te badane przez fizyków były dostosowywane tak, aby pasowały do modeli normalnych. Ich struktura częstotliwości różni się w zależności od użytego instrumentu i sposobu w jaki został zagrany dźwięk. Pomimo tego, że harmoniczne zawsze posiadają pewne odchylenie od perfekcyjnej harmonii (nie są dokładną algebraiczną wielokrotnością częstotliwości F0, a oscylują w jej okolicy) dla konkretnego instrumentu i sposobu zagrania są szczegółowo zdefiniowane przez pewien model matematyczny [33, s. 1326–1327]. Jak opisuje Manuel Davy w [28, s. 203–204] struktura muzyki tonalnej może zostać wykorzystana do budowy modeli bayesowskich, czyli modeli matematycznych w kontekście probabilistycznym, które prowadzą do najprostszego modelu wyjaśniającego daną falę dźwiękową. Pomimo tego, że modele te wydają się naturalnym rozwiązaniem estymacji wielotonowej, ponieważ rozwiązują problem dużej ilości parametrów w fali dźwiękowej, których nie można wyestymować bez pewnych założeń, nie są one często wykorzystywane. Przytoczona wcześniej praca [9] jest przykładem wykorzystania modeli bayesowskich do analizowania sygnałów muzycznych, w tym wykrywania F0.

#### 4.1. Wyniki algorytmów przeznaczonych dla sygnałów monofonicznych

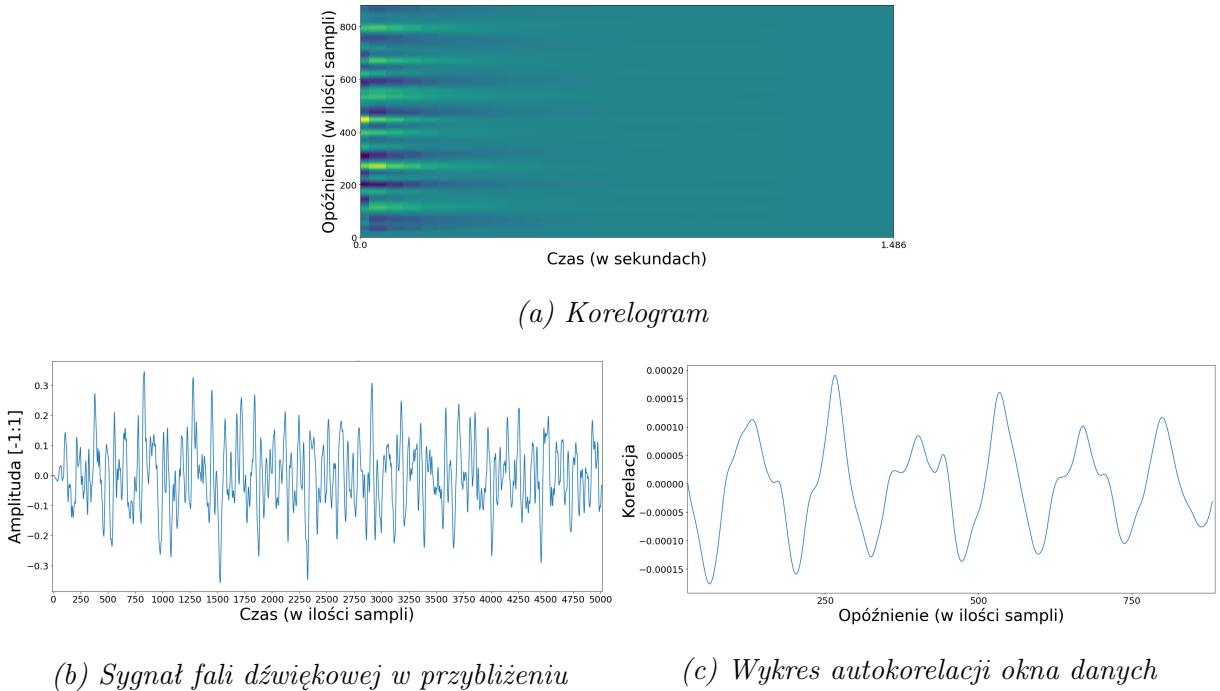
W rozdziale 3 struktura częstotliwości rozumiana była jako obraz pojedyńczego F0, z którego jednoznacznie wynika, jaka wysokość dźwięku jest grana w danym momencie czasu. Duża większość muzyki jest polifoniczna i tylko momentami skomplikowane kompozycje dopuszczają się brzmienia jedynie pojedynczych nut. W takich utworach metodologie transkrypcji muzyki monofonicznej są nieefektywne.



Rysunek 4.1: Funkcja autokorelacji 3.1 zastosowana na nagraniu Preludium op. 28 nr. 4 Fryderyka Chopina zagrany na pianinie w Em. Użyto okna Hanna o długości 2048 sampli z odstępami długości 2048 sampli.

Funkcja autokorelacji w domenie czasu/mocy, jak opisane zostało w sekcji 3.1, polegała na znalezieniu powtarzalności w fali dźwiękowej. Ta cykliczność była bezpośrednim efektem periodyczności pojedyńczych dźwięków. Gdy w tym samym czasie granych jest więcej nut, nawet na pojedyńczym instrumencie, AC nie jest w stanie odróżnić ich od siebie. Na rysunku 4.1 przedstawiony jest koreogram, wygenerowany analogicznie do koreogramu na rysunku 3.1c z tą różnicą, że sygnał bazowy jest tu polifoniczny. Nie można na podstawie rezultatu wnioskować żadnej z dźwięków granych w utworze.

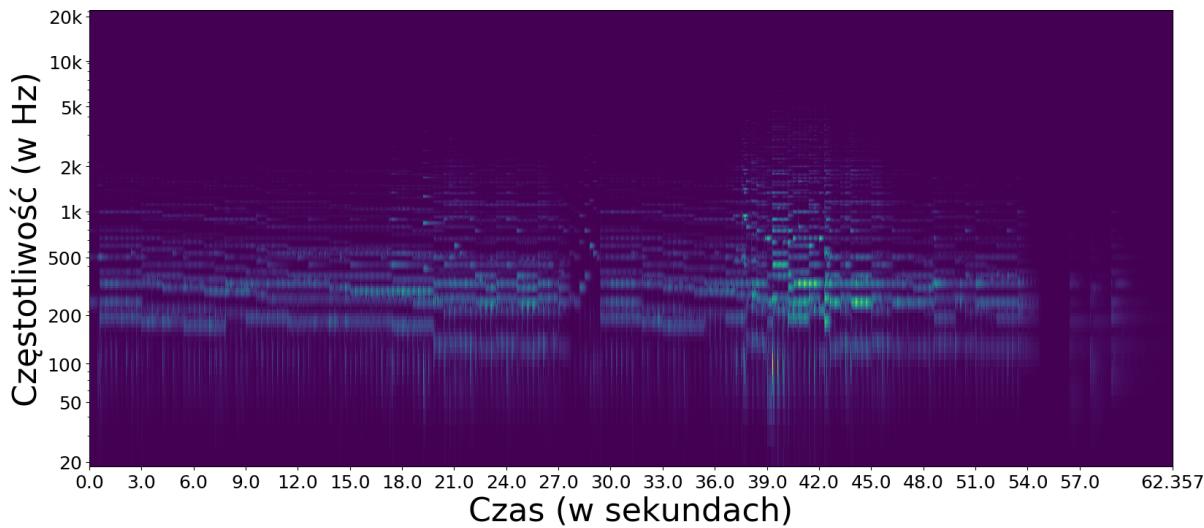
Przyglądając się z bliska pojedyńczemu wynikowi funkcji autokorelacji, jak widać na rysunku 4.2, można zauważyc, że pikи znajdują się w nieregularnych odstępach, a sama wartość funkcji korelacji nie jest bliska maksymalnej wartości. Na omawianej ilustracji grany jest akord E-mol 3, czyli dokładnie dźwięki E3, G3 i B3, którym odpowiadają częstotliwości F0 odpowiednio 164,81, 196,00 i 246,94 Hz. W badanym oknie korelacji



Rysunek 4.2: Wynik funkcji autokorelacji zastosowanej na nagraniu akordu Em zagrany na pianinie. Użyto okna Hanna o długości 2048 sampli z odstępami długości 2048 sampli. Wykresy (c) jest reprezentacją wyników AC na 11 oknie (próbce danych od 20480 do 22528 sampli sygnału wejściowego).

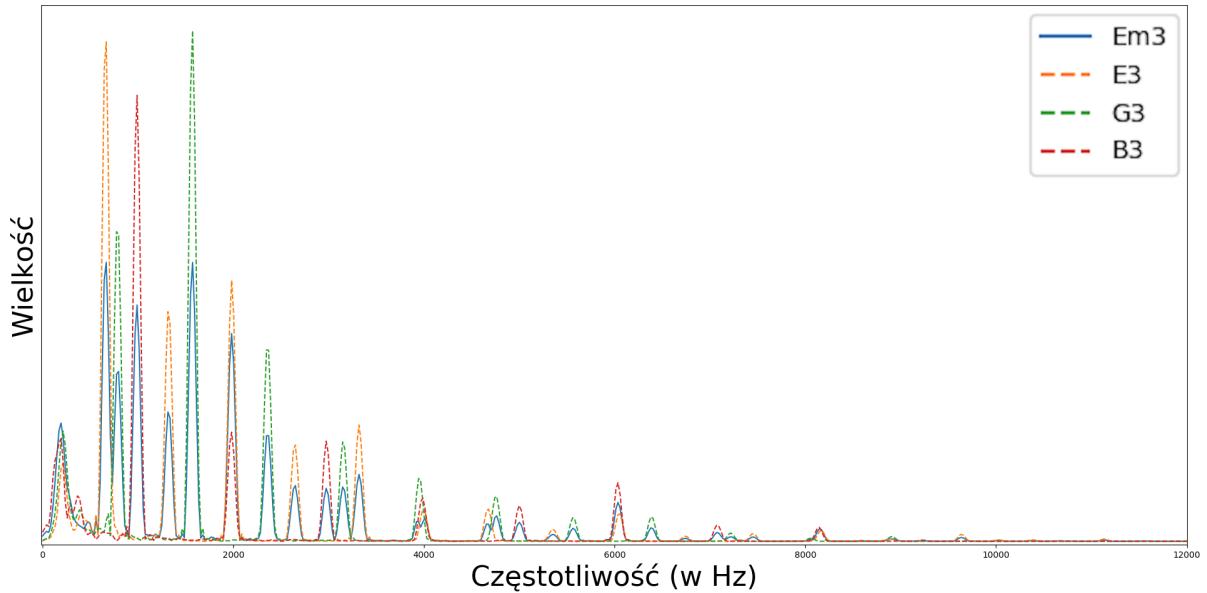
najwyższy pik jest w miejscu o przesunięciu 266, czyli najsilniejszą hipotezą co do F0 byłaby częstotliwość 165,79 Hz (zgodnie ze wzorem 3.3 w badanym przypadku częstotliwość wyliczana jest poprzez  $44100/266 \approx 165,79$ ), co jest wartością najbliższą do częstotliwości dźwięku E3. Najniższa nuta w akordzie nazywana jest bazową, i to ona jest najbardziej charakterystyczna dla ludzkiego ucha. Z racji tego, że ma ona najwyższą częstotliwość, ma też najwyższą amplitudę (z reguły im wyższa częstotliwość tym niższa amplituda), co daje wyższą szansę na znalezienie tych nut w dźwięku przy wykorzystaniu funkcji autokorelacji w domenie czasu. Drugi pik pod względem rezultatu autokorelacji jest dla przesunięcia równego 534, co odpowiada częstotliwości równej 52,6 Hz. Ta częstotliwość nie jest częścią granego dźwięku, nie jest również żadną z jej wielokrotności.

Metody operujące na dziedzinie częstotliwości mają dużo większą skuteczność w wykrywaniu równoległych F0, jednak algorytmika stojąca za tymi metodami przeznaczonymi do analizy sygnałów polifonicznych są dużo bardziej skomplikowana niż te, w przypadku monofonicznych utworów. Dzieje się tak ze względu na wyższy poziom złożoności



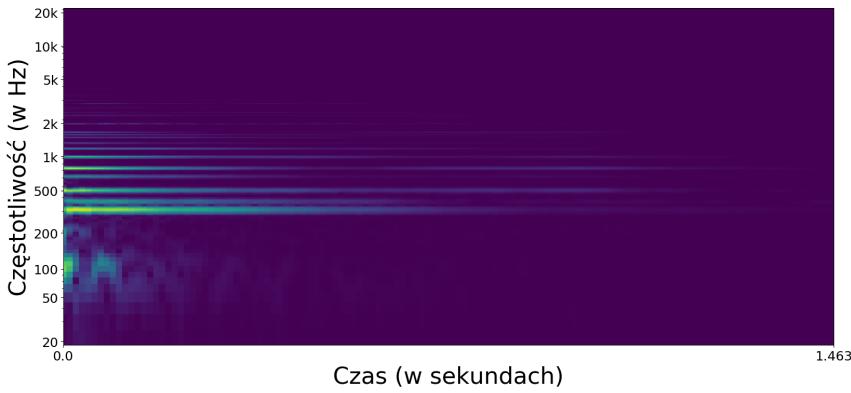
*Rysunek 4.3: Spektrogram nagrania Preludium op. 28 nr. 4 Fryderyka Chopina w Em zagraneego na pianinie. Użyto okna Hann'a o długości 2048 sampli, marginesu zer o długości 2048 i odstępów długości 512 sampli. Jaśniejsze obszary w spektrogramie oznaczają większą magnitudę.*

widma. Spektrogram sygnałów polifonicznych nie jest tak jednoznacznym wskaźnikiem jak był w przypadku sygnałów monofonicznych. Częstotliwości harmonicznych mogą się na siebie nakładać, zniekształcając ich cykliczność jak i modulując wyraźność formantów. Przykład takiego zjawiska można zaobserwować na rysunku 4.4. Nagranie akordu Em3 zagraneego na pianinie widać linią ciągłą, natomiast pojedyńczo nagrane dźwięki E3, G3 oraz B3 są przedstawione jako osobne krzywe. Na wykresie widoczne jest, że wynikowa krzywa Em żadko nakłada się dokładnie z linią pojedyńczej nuty, a jest raczej nową strukturą z częścią cech wspólnych.

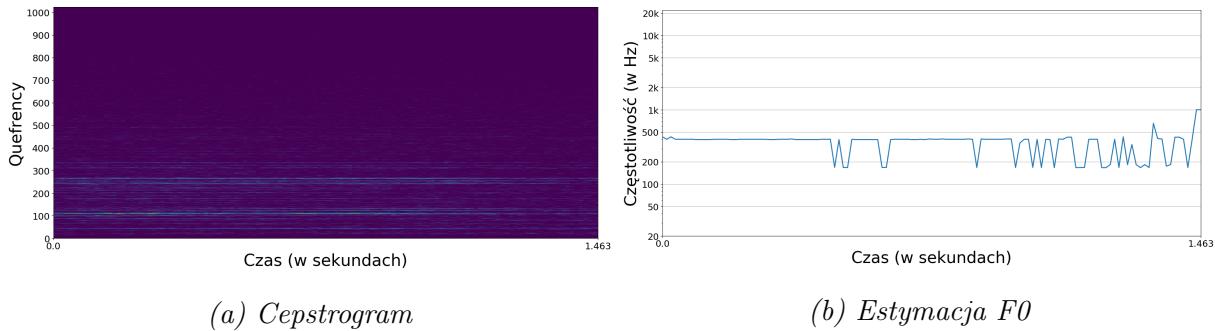


Rysunek 4.4: Spektralne komponenty nagrania akordu Em z pianinie oraz trzech poszczególnych nut zagranych na tym samym pianinie nagranych osobno. Użyto okna Hanna o długości 2048 sampli i marginesu zer długości 2048.

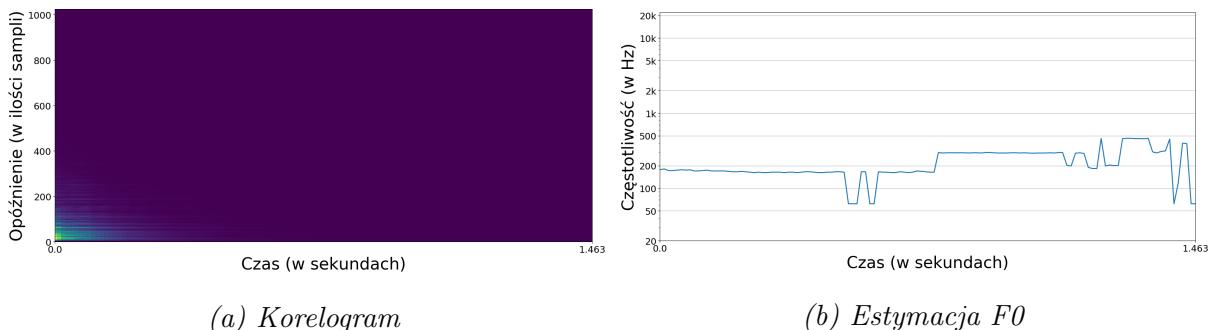
Metody cepstrum i ACLOS opisane w sekcjach 3.3 i 3.4 są nieefektywne z powodu nakładających się na siebie dźwięków w spektrum. Echo, które wykrywa cepstrum, jest pomieszane pomiędzy dźwiękami, co mocno komplikuje wnioskowanie z tej reprezentacji. ACLOS natomiast traci pewność wyniku z tych samych powodów co funkcja autokorelacji opisana wyżej w tej sekcji. Można zaobserwować na wykresach 4.6b i 4.7b, że oba te algorytmy wykryły F0 w okolicach dźwięku G (ACLOS około 190 Hz, co jest najbliższe dźwiękowi G3 obecnemu w sygnale, a Cepstrum około 400, co jest wielokrotnością G3). Przyglądając się rozbitemu na poszczególne spektra wykresowi 4.4 można wywnioskować, że nuta ta ma najczęściej wspólnych harmonicznych z pozostałymi, więc właśnie ona jest najsilniejsza w widmie.



Rysunek 4.5: Spektrogram nagrania akordu Em zagraneego na pianinie. Użyto okna Hanna o długości 2048 sampli z marginesem zer długości 2048 i marginesu zer długości 2048. Jaśniejsze obszary w spektrogramie oznaczają większą magnitudę.



Rysunek 4.6: Wynik analizy cepstralnej nagrania akordu Em zagraneego na pianinie. Użyto okno Hanna o długości 2048 sampli z marginesem zer długości 2048 i odstępami długości 2048 sampli. Jaśniejsze obszary w cepstrogramie oznaczają większą magnitudę.



Rysunek 4.7: Wynik analizy algorytmem ACLOS nagrania akordu Em zagraneego na pianinie. Użyto okna Hanna o długości 2048 sampli z marginesem zer długości 2048 i odstępami długości 2048 sampli. Jaśniejsze obszary w spektrogramach oznaczają większą magnitudę.

## 4.2. Metody oparte na modelach generatywnych

W książce [28, s. 203–227] autorzy kategoryzują metodyki przeznaczone do transkrypcji częstotliwości fundamentalnych opartych na modelach generatywnych fal akustycznych ze względu na charakter algorytmu jaki jest użyty. Modele generatywne fal akustycznych czerpią z modeli bayesowskich. Rozróżnili oni metody oparte na zaszumione modele sum sinusoidów, które nie brały pod uwagę zależności pomiędzy harmonicznymi (przykładem takiego podejścia jest analiza mowy opisana w [32, s. 744–745]), oraz grupy modeli off-line i on-line.

Podejścia off-line charakteryzują się globalnym mierzeniem parametrów estymacji. Operują one na analizie całej fali dźwiękowej. Założeniem tych algorytmów jest segmentacja danych w taki sposób, że każdy z segmentów ma niezmienne wysokości granicznych dźwięków. Ograniczenie to wymusza na systemie transkrypcji użycia jakiejś formy wykrywania początków dźwięków aby móc podzielić dane wejściowe według tego założenia. Algorytmy te bazują również na pojęciu prawdopodobieństwa, które to opisane jest matematycznym modelem wraz z gęstością. Algorytmy w tej grupie często działają według pewnych założeń co do barwy dźwięków (instrumentów czy głosu) występujących w sygnale.

Metodologie on-line są znacznie większą grupą w sensie ilości i zróżnicowania algorytmów. Dla każdego kroku procesu  $n$  badana jest tylko część sygnału (okno)  $x(n)$ . Wyniki ze wcześniejszych okien są również brane pod uwagę - jako rozkład prawdopodobieństwa przejścia. Ważną różnicą pomiędzy podejściami on-line i off-line jest to, że te pierwsze nie potrzebują dodatkowego mechanizmu wykrywającego początki dźwięków. Autorzy w [28, s. 203–227] rozróżnili modele on-line również po tym, czy tworzony model bezpośrednio miał determinować kształt fali, czy też być modelem pośrednim w bardziej skomplikowanym systemie. Dokładniej, opisuję cztery podejścia stosowane dla ogólnego przetwarzania audio, z czego wszystkie wykorzystują cechę równomiernego rozłożenia harmonicznych akustycznej fali dźwiękowej.

W tej sekcji przybliżone są metody Antonio Pertusa i Jose M. Iñesta dotyczące automatycznej transkrypcji polifonicznego sygnału fonicznego przy pomocy równoległych metod w [40] i [39] wraz z metodologiami opartymi na harmonicznosci i gładkości widma opisanymi w pracy Anssi Klapuri [27] i pracy Chunghsin Yeha i Axel Roebela [51].

### 4.2.1. Przetwarzanie wstępne

W pracy [51, s. 1] Yeh i Röbel zakładają, że quasi-harmoniczny, polifoniczny sygnał można przedstawić za pomocą poniższego wzoru:

$$y[n] = \left\{ \sum_{m=1}^M \sum_{h_m=1}^{H_m} a_m, h_m[n] \cos((1 + \delta_{m,h_m}) h_m \omega_m n + \phi_m[n]) \right\} + v[n], \quad (4.1)$$

gdzie  $n$  jest indeksem czasu,  $M$  jest liczbą źródeł dźwięków,  $H_m$  jest liczbą części dla  $m$ -tego źródła,  $\omega_m$  reprezentuje F0 źródła  $m$  a  $\phi_m[n]$  jest fazą. Tak zarysowany sygnał jest w dalszej kolejności analizowany w oparciu o poszczególne komponenty. Warto zwrócić uwagę na to, że zakłada się szum o na tyle małej amplitudzie, że możliwym jest znalezienie poszczególnych sinusoidów składowych.

Innym istotnym wzorem opisującym sygnał źródłowy jest jego reprezentacja w domenie częstotliwości. W swojej pracy Anssi Klapuri [27, s. 806] zdefiniował kształt spektrum takiego sygnału jako funkcję częstotliwości:

$$X(k) = H(k)S(k) + N(k) \quad (4.2)$$

gdzie  $X(k)$  jest dyskretnym spektrum mocy akustycznego sygnału źródłowego,  $S(k)$  jest spektrum mocy wibrującego systemu, którego częstotliwość fundamentalna jest mierzona,  $H(k)$  jest charakterystyką środowiska, w którym dźwięk został wygenerowany wraz z fizycznymi charakterystykami instrumentu, który filtryuje źródło dźwięku i wszystkimi innymi czynnikami wpływającymi na modulacje nagrywanego sygnału a  $N(k)$  jest spektrum mocy szumu addytywnego.

Generalizując, proces przygotowujący sygnał do transkrypcji muzyki można opisać jako eliminację  $H(k)$  oraz  $N(k)$  zgodnie ze wzorem 4.2. Szum  $N(k)$  można tłumić poprzez odejmowanie w dziedzinie spektrum mocy. Proces eliminacji  $H(k)$  nosi nazwę wstępne wybielanie i można go dokonać poprzez lifting wysoko przepustowy (filtrowane są niższe wartości od zadanej w domenie quefrencji). Odruchowym byłoby użycie obu tych metod, aby w wyniku dostać możliwie dokładne  $S(k)$ , jednak zgodnie z badaniami opisanymi w [19, s. 84–86] kaskadowe wykonywanie algorytmów redukujących szum i modulacje akustyczne nie polepsza wyniku działania algorytmów przetwarzających tak przygotowany sygnał. Klapuri w [27, s. 806] przedstawia metodę krzywienia magnitudy,

która wyrównuje  $H(k)$  i jednocześnie pozwala na liniowe odjęcie szumu w kaskadowy sposób. Opisane wykrzywianie dla spektrum mocy  $X(k)$  zdefiniowane jest wzorem:

$$Y(k) = \ln\left\{1 + \frac{1}{g} X(K)\right\} \quad (4.3)$$

gdzie  $g$  opisane jest wzorem:

$$g = \left[ \frac{1}{k_1 - k_0 + 1} \sum_{l=k_0}^{k_1} X(l)^{\frac{1}{3}} \right]^3 \quad (4.4)$$

Wartości  $k_0$  i  $k_1$  odpowiadają odpowiednio minimalnej i maksymalnej częstotliwości, jaka jest dopuszczalna w algorytmie estymującym F0. Jednym z założeń tego podejścia jest znacznie większa amplituda  $H(k)S(k)$  od amplitudy szumu  $N(k)$ . Drugim założeniem jest to, że większość komponentów spektralnych na odcinku od  $k_0$  do  $k_1$  ma moc na poziomie szumu, a jedynie kilka komponentów jest faktycznymi pikami. Z tymi założeniami zadaniem mnożenia przez  $\frac{1}{g}$  jest skalowanie sygnału wejściowego tak, że szum zostaje na równym poziomie, a piki są wyraźnie widoczne ponad tym poziomem. Wzór 4.4 nakłada na szum liniową transformację krzywienia magnitudy, podczas gdy piki w widmie przechodzą transformację logarytmiczną. Powszechnie jest również eliminowanie szumów  $N(k)$  poprzez branie pod uwagę widma tylko ponad pewnym ustalonym progiem tak, aby wszystkie komponenty poniżej tego progu mogły być uznane za nieistotne. Zakłada się też, że charakterystyka otoczenia  $H(k)$  ma na widmie na tyle mały efekt, że można ją pominać [40]. Założenie to ogranicza jakość nagrani muzycznych branych pod uwagę w kontekście badania F0 do tych, które zostały nagrane profesjonalnie (sprzętem zaniedbywalnie modyfikującym charakterystykę granych instrumentów, w pomieszczeniu o neutralnej w kontekście modulacji fal dźwiękowych akustyce), aby zminimalizować wpływ  $H(k)$ .

#### 4.2.2. Modelowanie obwiedni spektrum

Według zasady przytoczonej wcześniej, wysokie częstotliwości w sygnale mają w ogólności mniejszą magnitudę. Własność ta powoduje, że kształt widma sygnału akustycznego ma postać rozkładu prawoskośnego. Ta cecha spektrum jest niezwykle użyteczna w kontekście transkrypcji muzyki.

Przy założeniu, że częstotliwość fundamentalna F0 ma  $n$  harmonicznych w badanym spektrum, oznaczonych kolejno  $f_0, f_1, f_2, \dots, f_n$ , zasada *gładkości spektrum* mówi, że magnitudy kolejnych harmonicznych sygnału pozostają w następującej relacji:

$$m(f_0) > m(f_1) > m(f_2) > \dots > m(f_n) \quad (4.5)$$

gdzie  $m(f_n)$  jest magnitudą  $n$ -tej harmonicznej.

W ogólności, wykorzystanie gładkości spektrum w procesie transkrypcji polega na założeniu, że obwiednia widma sygnału akustycznego jest relatywnie gładka i wraz ze wzrostem częstotliwości obserwowanego komponentu spektralnego zmniejsza się jego moc. W przypadku, gdy w spektrum istnieje skok w magnitudzie pomiędzy sąsiednimi harmonicznymi danego kandydata F0, który zakłóca kształt widma, można podejrzewać, że skok ten jest wynikiem nałożenia się na siebie harmonicznych różnych F0.

Sytuacja, w której część harmonicznych dwóch różnych częstotliwości fundamentalnych nakłada się na siebie została opisana przez Anssi Klapuri w [26, s. 3382] jako  $hF_S = jF_R$ , gdzie  $S$  i  $R$  to różne źródła równoległych dźwięków,  $F_S$  i  $F_R$  to ich częstotliwości fundamentalne a  $h$  i  $j$  oznaczają numery harmonicznych dla danych F0. Po redukcji czynników wspólnych liczb całkowitych  $h$  i  $j$  wzór wygląda następująco:

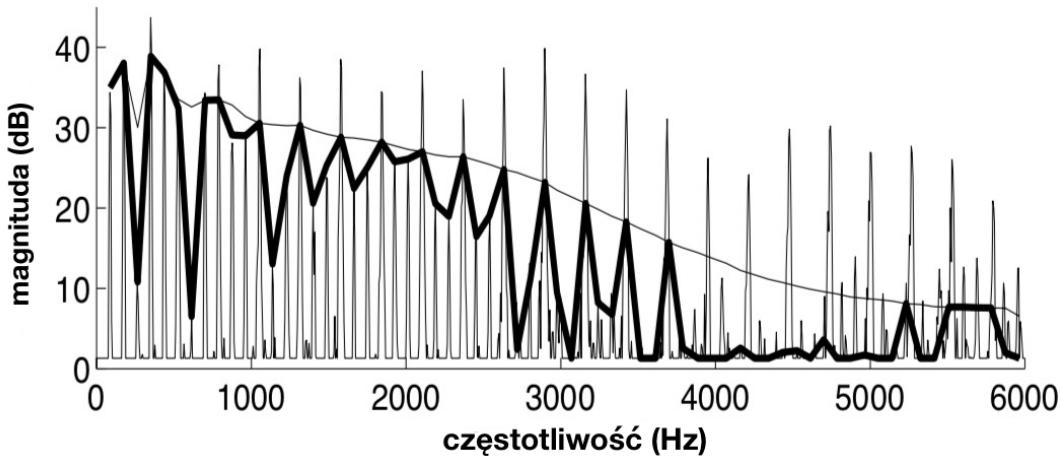
$$F_R = \frac{m}{n} F_s \quad (4.6)$$

gdzie  $(m, n) \geq 1$  są liczbami całkowitymi. Zgodnie z równaniem 4.6, częstotliwości harmoniczne dwóch różnych F0 mogą się pokrywać nawzajem tylko wtedy, kiedy częstotliwości fundamentalne są między sobą w relacji wymiernej. Jak opisuje Anssi Klapuri w [26, s. 3382], gdy dwa dźwięki są w opisanej wyżej relacji, to każda  $m$ -ta harmoniczna  $mk$  dźwięku  $S$  nakłada się z  $n$ -tą harmoniczną  $nk$  dźwięku  $R$ , gdzie  $k \geq 1$ . Ścisłe ujmując,  $hF_S = jF_R$  dla każdej pary  $h = mk$  i  $j = nk$  gdy 4.6 jest prawdziwe.

Anssi Klapuri kategoryzuje trudności związane z relacjami wysokości dźwięków na dwie grupy:

- W trakcie usuwania wykrytego dźwięku ze spektrum, harmoniczne innego dźwięku mogą być usunięte, co spowoduje utrudnienie albo nawet uniemożliwienie wykrycia tego dźwięku,
- Dwie lub więcej F0 są ze sobą w takiej relacji, że w widmie pojawia się nieistniejący dźwięk „duch”. Dzieje się tak np. gdy dwa F0, o częstotliwościach odpowiednio  $2F$  i  $3F$  są grane równolegle, harmoniczne obu dźwięków nakładają się z co drugimi i trzecimi harmonicznymi nie granego dźwięku o częstotliwości F0 równej  $F$ . Tak powstały dźwięk może zostać niewłaściwie wykryty przez algorytmy transkrypcji.

Trudności te mają być zminimalizowane poprzez nałożenie funkcji wygładzającej na dane spektrum. Przykład przedstawiony na rysunku 4.8 obrazuje funkcję wygładzenia



Rysunek 4.8: Zobrazowanie zasady gładkości spektrum. Logarytmiczne spektrum mocy zawierające dwa dźwięki. Widmo zostało przefiltrowane przez lifting wysoko przepustowy. Wykres z [26, s. 3383].

(cieńka linia) nałożoną na spektrum podczas badania pierwszego dźwięku, które obrazuje, że co trzeci pik jest wysoko ponad zakładaną krzywą, i to one zostałyby w widmie po odejęciu wygładzonego widma pierwszego dźwięku ze spektrum sygnału. Ten sam mechanizm może pozwolić na wyeliminowanie nieistniejących dźwięków „duchów”.

Gładkość spektrum jest założeniem, które pozostaje prawdziwe dla większości sygnałów akustycznych, lecz nie jest to cecha, którą posiadają wszystkie dźwięki. W zależności od źródła vibracji, otoczenia i sposobu, w jaki sygnał jest nagrany, charakterystyka widma może odbiegać od tej, którą zakłada gładkość spektrum. Silniejsze formanty występujące w wyższych częstotliwościach są cechą wielu instrumentów akustycznych. W celu zminimalizowania błędów spowodowanych założeniem gładkości spektrum powstała kategoria algorytmów bazujących na wyuczonych modelach dźwięków. Zakładają one istnienie modeli dla każdego ze źródeł dźwięków, które mogą występować w analizowanym sygnale, które to są wykrywane w widmie. Problemem w takim podejściu jest ilość potrzebnych danych - w muzyce zachodniej wykorzystywanych jest bardzo dużo różnych instrumentów, z bardzo odległymi od siebie charakterystykami widmowymi. Potrzebne dane treningowe, poprawnie oznaczone według wysokości F0 oraz użytego instrumentu na skale ogólnej analizy popularnej muzyki zachodniej są praktycznie niemożliwe do usystematyzowania. Jednym z ułatwień jest wykorzystanie pojedyńczego nagrania F0 na danym instrumencie i przesuwanie go w wymiarze częstotliwości, aby symulować inne wysokości F0, ale nie

jest to niezawodne rozwiązańe, ponieważ wiele instrumentów ma różne charakterystyki widmowe podczas grania różnych częstotliwości fundamentalnych [42, s. 55–68].

#### 4.2.3. Nieharmoniczność

Wiele algorytmów zaprojektowanych do transkrypcji muzyki bierze pod uwagę problem nieharmoniczności dźwięku. Jak opisuje Anssi Klapuri w pracy [27, s. 806–808], fizyczne źródła dźwięków wytwarzają harmoniczne, które nie posiadają dokładnych częstotliwości będącymi wielokrotnościami F0, lecz jedynie oscylują w ich okolicy. Dla przykładu, rozbieżność tą dla instrumentów strunowych można opisać jako:

$$f_h = hF\sqrt{1 + (h^2 - 1)\beta} \quad (4.7)$$

gdzie  $F$  jest częstotliwością fundamentalną a  $\beta$  jest współczynnikiem nieharmoniczności [27, s. 807].

W przytoczonej wcześniej pracy [9, s. 1–10] Davy i Godsill zaproponowali model matematyczny, który bierze pod uwagę nieharmoniczność (nazywaną w tej pracy jako nie-nastrojenie, z ang. *de-tuning*). Opisany w tej pracy jest model zarówno dla sygnałów monofonicznych, jak i dla polifonicznych. Ten drugi zapisany jest jako dyskretnie obserwacje  $y$  w domenie czasu  $t$  w następujący sposób:

$$y_t = \left\{ \sum_{k=1}^K \sum_{m=1}^{M_k} \sum_{i=1}^I a_{k,m,i} \phi_{i,t} \cos[(m + \delta_{k,m})\omega_{0,k}t] + b_{k,m,i} \phi_{i,t} \sin[(m + \delta_{k,m})\omega_{0,k}t] \right\} + v_t \quad (4.8)$$

dla  $t = 0, 1, \dots, N - 1$ , gdzie  $K$  jest liczbą równolegle granych dźwięków w momencie  $t$ ,  $M$  to ilość występujących harmonicznych dla danego dźwięku,  $\omega_{0,k}$  to częstotliwość fundamentalna dźwięku  $k$ , nieharmoniczność przedstawiona jako  $\delta = [\delta_1, \dots, \delta_K]$ , gdzie  $\delta_k = [\delta_{k,1}, \dots, \delta_{k,M_k}]^T$ .  $I$  jest stałe, znane, i jego celem jest reprezentacja amplitudy jako gładkiej funkcji bazowej  $\theta = [a_{1,1,1}, b_{1,1,1}, \dots, a_{K,M_K,I}, b_{K,M_K,I}]^T$ . Można zauważać podobieństwo do wzoru 4.1 przedstawionego przez Yeh'a i Röbel'a. Autorzy 4.8 opisują parametr nieharmoniczności jako:

„Wektor drobnego de-nastrojenia harmonicznych  $\delta$ . Jego wartość powinna być bliska zeru i bez relacji pozostałych harmonicznych względem  $\delta$ , pomimo tego, że instrument w teorii determinuje rozkład kolejnych harmonicznych. W tym przypadku zakłada się zerową średnią niezależność Gaussa, z wariancją  $\sigma_\delta^2$ , ustalony dla małych wartości aby faworyzować małe parametry de-nastrojenia. Rozkład można

dodatkowo obciążć tak, aby sąsiednie harmoniczne nie krzyżowały się w domenie częstotliwości” [9, s. 9] (tłumaczenie własne).

#### 4.2.4. Metoda Pertusa i Iñesta (2008)

Metoda, którą opracowali razem Antonio Pertusa i José M. Iñesta w 2008 roku nosi nazwę „Estymacja wielu częstotliwości fundamentalnych z użyciem gładkości Gaussa” (tłumaczenie własne) [40]. Jest to metoda, wykorzystująca okno Gaussa do wygładzenia widma sygnału i generującą listę hipotez, z której dalej determinowany jest wynik transkrypcji. Algorytm ten przetwarza dane przed faktyczną analizą, jak zostało to opisane w sekcji 4.2.1, wykorzystuje zasadę gładkości spektrum opisaną w sekcji 4.2.2 i bierze pod uwagę spektralną nieharmonicznych, której problem opisany był w sekcji 4.2.3.

Algorytm wykonuje estymację F0 operując na spektrum kolejnych okien czasowych. Analizowane fragmenty sygnału wejściowego przygotowane są tak, aby miały długość 93 ms (4096 sampli przy częstotliwości samplowania równej 44,1 kHz) z zastosowaniem funkcji Hanninga jako funkcji okna oraz marginesu zer równego trzykrotności długości oryginalnego okna. Na tak przygotowanym oknie wykonywane jest STFT, którego wynik ma 16384 spektralnych komponentów, więc każdy z nich ma pasmo szerokości 2,7 Hz (szerokość pasma jest liczona wzorem  $h = Fs/N$ , gdzie  $Fs$  jest częstotliwością próbkowania, a  $N$  długością okna).

Dla każdego okna wyniku STFT wykonywane jest przetwarzanie wstępne. Polega ono na odrzuceniu wszystkich wyników, których amplituda jest poniżej wyznaczonego progu  $\mu$ . W implementacji tego kroku wprowadzona została modyfikacja, która zakłada, że tak odfiltrowane wyniki są tylko pikami. Z tym założeniem, gdy w danym zakresie jest wiele komponentów o amplitudzie wyższej niż  $\mu$ , zachowywany jest tylko jeden, o najwyższej amplitudzie. Dokładniejszy opis znajduje się w sekcji 5.4.

W kolejnym kroku ze spektralnych pików wybierany jest zbiór częstotliwości nazywany „kandydatami F0”. Pik jest uznawany za kandydata jeżeli jego częstotliwość jest w zadanym zakresie  $[f_{min}, f_{max}]$  oraz posiada on w danym spektrum co najmniej  $\eta$  harmonicznych. Pierwszy z warunków ogranicza grupę analizowanych F0 do jedynie tych, które mogą występować w kompozycji. Badanie skrajnie niskich lub skrajnie wysokich w tonie dźwięków jest niepotrzebne, ponieważ nie są one wykorzystywane w muzyce. Drugie założenie ogranicza grupę kandydatów odrzucając tych, których barwa jest niemożliwa do określenia. Aby wykryć harmoniczne danego F0 sprawdzane jest istnienie pików o czę-

stotliwości będącej wielokrotnością częstotliwości kandydata F0. W związku z istnieniem nieharmoniczności, badane są częstotliwości w zasięgu ze stałym marginesem  $hf_0 \pm f_r$  dla  $h = 2, 3, \dots$ , gdzie  $f_0$  jest kandydatem F0. Wybierany jest pik najbliższy wielokrotności  $f_0$ . W przypadku nie znalezienia żadnego piku w danym zakresie, harmoniczna uznawana jest za brakującą. W następnym kroku lista kandydatów sortowana jest względem sumy amplitud ich harmonicznych. Dodatkowym parametrem jest maksymalna liczba kandydatów  $F$ , na podstawie której odrzucani są nadmiarowi kandydaci o najmniejszej sumie amplitud, aby zwiększyć wydajność tego systemu transkrypcji.

Na podstawie kandydatów generowane są wszystkie możliwe ich kombinacje, a następnie są wyliczane dla nich współczynniki „istotności” (z ang. *salience*) oznaczone dalej jako  $S$ . Kombinacja, która ma największą wartość tego współczynnika, jest wybierana, co oznacza, że w danym oknie czasowym wszyscy kandydaci F0 danej kombinacji są obecni w kompozycji. Aby zredukować złożoność obliczeniową tego kroku, generowane są kombinacje z maksymalnie  $C$  kandydatami, co przekłada się na maksymalnie  $C$  równoległych nut granych w tym samym czasie w analizowanym utworze.

Istotnością kombinacji jest suma współczynnika istotności każdego z jej kandydatów. Jako istotność brane są pod uwagę amplitudy poszczególnych harmonicznych F0 wyliczane na podstawie gładkości ich widm. Dla pojedynczego kandydata  $c$  wyznaczany jest wektor „wzoru widmowego”  $p$ , który zawiera informacje o amplitudach każdej z harmonicznych kandydata. W literaturze wektor ten powszechnie określany jest jako HPS (z ang. *hypothetical partial sequences*) czyli hipotetyczne częściowe sekwencje (tłumaczenie własne) [52, s. 1118]. Na podstawie tych wektorów wykonywany jest iteracyjny algorytm, którego celem jest oznaczenie przynależności każdej harmonicznej do kandydatów występujących w danej kombinacji. Kategoryzuje on harmoniczne na dwie grupy: te, które są w widmie tylko jednego kandydata i te, które występują w widmach kilku kandydatów (i muszą zostać „wygładzone”).

Wyliczanie istotności odbywa się dla każdego kandydata osobno, zaczynając od tych z najniższą częstotliwością. Na podstawie danego wzoru widmowego  $p_c$  dla kandydata  $c$  wszystkie harmoniczne, które występują w więcej niż jednym widmie, są liniowo interpolowane przy użyciu harmonicznych, które nie są dzielone pomiędzy kandydatów. Po tej operacji istnieją dwie możliwości:

1. interpolowana wartość jest większa lub równa amplitudzie dzielonej harmonicznej. W

- tym przypadku amplituda uwzględniana przy liczeniu istotności kandydata  $c$  dla tej harmonicznej jest brana ze wzoru widmowego  $p_c$ , natomiast we wszystkich pozostałych wektorach  $p$ , w których występuje ta harmoniczna, amplituda jej jest zerowana,
2. interpolowana wartość jest mniejsza od amplitudy dzielonej harmonicznej. W tym przypadku interpolowana wartość przypisywana jest do wzoru widmowego  $p_c$ , a w pozostałych wektorach, w których występuje ta harmoniczna, amplituda jej jest pomniejszana o interpolowaną wartość.

Proces ten ma na celu wygładzenie spektrum dla każdego z kandydatów  $c$ , zgodnie z zasadą gładkości opisaną w sekcji 4.2.2.

Głośność  $l(c)$  każdego kandydata  $c$  występującego w danym wzorcu widmowym  $p$  wyliczana jest poprzez zsumowanie amplitud występujących w  $p_c$

$$l(c) = \sum_{h=1}^H p_{c,h} \quad (4.9)$$

Wyliczany jest również współczynnik gładkości  $\sigma$  dla każdego  $c$  z użyciem gładkości Gaussa. Wektor  $p_c$  jest filtrowany dolno-przepustowo przy pomocy obciętego znormalizowanego okna Gaussa z trzema komponentami  $N_{\sigma=1.0} = \{0, 21, 0, 58, 0, 21\}$ , które nakładane jest na wzorzec spektralny danego kandydata tworząc wygładzony wzorzec  $\tilde{p}$ . Mając wygładzony HPS, wyliczany jest współczynnik ostrości danego kandydata  $s(c)$  jako suma wartości bezwzględnej różnicy wygładzonego i oryginalnego wzorca. Wynik tej operacji normalizowany jest poprzez wyliczenie ilorazu z ilością harmonicznych w wektorze  $p_c$ :

$$\bar{s}(c) = \frac{1}{H} \sum_{h=1}^H (|\tilde{p}_{c,h} - p_{c,h}|) \quad (4.10)$$

Tak wyliczony współczynnik ostrości wykorzystuje się do obliczenia współczynnika gładkości  $\sigma(c)$ , który jest przeciwnością  $\bar{s}(c)$ :

$$\sigma(c) = 1 - \bar{s}(c) \quad (4.11)$$

Istotność wyliczana jest na podstawie współczynników głośności i gładkości każdego z kandydatów jako:

$$S = \sum_{c=1}^C [l(c) * \sigma(c)]^2 \quad (4.12)$$

gdzie  $C$  jest ilością kandydatów danej kombinacji. Kombinacja odrzucana jest, gdy różnica wartości współczynników głośności pomiędzy dwoma dowolnymi jej kandydatami przekracza ustaloną wartość  $\gamma$ . Jest to zabezpieczenie przed wyborem kombinacji, w których

istnieje grupa kandydatów o zaniedbywalnej głośności. W takim wypadku powinna zostać wybrana inna kombinacja.

W pracy opisującej algorytm autorzy zaproponowali zastosowanie przetwarzania końcowego, które ma na celu zminimalizowanie ilości błędów w transkrypcji [40, s. 106–107]. Składa się ono z dwóch operacji:

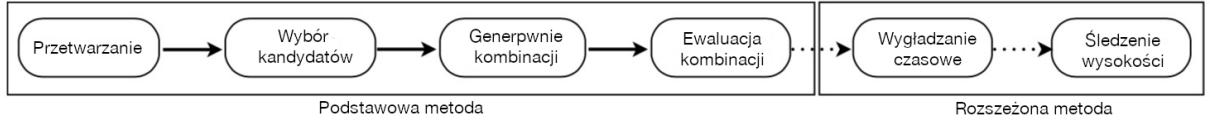
1. założenie znalezienia nuty, jeżeli nie była wykryta w danym oknie czasowym, ale była wykryta w poprzednim i kolejnym oknie (np. w przypadku, gdy dźwięk C3 był wykryty w oknach o indeksie 10 i 12, ale nie był wykryty w oknie o indeksie 11, to zakłada się, że w oknie o indeksie 11 również wykryto ten dźwięk),
2. usunięcie bardzo krótkich dźwięków. W pracy zasugerowano minimalną długość sześciu okien (przy założeniach wstępnych częstotliwości próbkowania i długości okna wynosi to 55.68 ms).

Tak przetworzone dane są zapisywane do pliku MIDI na podstawie informacji o początku dźwięku, jego długości oraz wysokości.

#### 4.2.5. Metoda Pertusa i Iñesta (2012)

Opisana w sekcji 4.2.4 metoda do równoległej estymacji F0 w każdym z okien czasowych została po czterech latach rozszerzona na podstawie nowych dokonań w dziedzinie transkrypcji muzyki i opisana przez tych samych badaczy w pracy [39]. Założenia metody pozostają bez zmian - dla każdego badanego okna wyliczane są współczynniki istotności w oparciu o kombinacje kandydatów F0, a następnie wybierana jest kombinacja o największym współczynniku  $S$ . Różnicę od pierwotnej metodologii, jakie narzuca znormalizowany algorytm, składają się z aktualna sposób wyliczania istotności o normalizację współczynnika gładkości oraz dodania zaawansowanych technik przetwarzania końcowego dla każdego okna w oparciu o okna sąsiednie. Ogólny schemat działania tej metody jest przedstawiony na rysunku 4.9.

Dla danej kombinacji kandydatów częstotliwości fundamentalnych  $c$  wyliczanie parametru wykonywane jest tak samo jak w podstawowej metodzie, tj. zgodnie ze wzorem 4.9. Odrzucane są kombinacje, które posiadają kandydata z głośnością poniżej wyznaczonego progu ( $l(c) < \eta$ ) lub takie, w których różnica pomiędzy najchętszym i najgłośniejszym kandydatem jest wystarczająco duża ( $l(c) < \gamma L_C$ , z  $L_C$  będącym głośnością najbliższego kandydata).



Rysunek 4.9: Schemat metody z podziałem na metodę podstawową opisaną w [40] i rozszerzoną opisaną w [39]. Schemat z [39, s. 3], tłumaczenie własne.

Istotna zmiana pojawia się podczas liczenia parametru gładkości danego kandydata. Wektor HPS  $p_c$  danego kandydata jest poddawany normalizacji poprzez podzielenie jego wartości przez jego maksymalną wartość, skalując go do przedziału  $\langle 0 - 1 \rangle$ . Tak znormalizowany HPS jest oznaczany jako  $\bar{p}$ . Kolejno wzory na wygładzony wektor wzoru widmowego i współczynnik ostrości danej kombinacji są wyliczane z użyciem znormalizowanego HPS:

$$\tilde{p} = N_{0,1} * \bar{p}_c \quad (4.13)$$

$$r(c) = \sum_{h=1}^H |\tilde{p}_{c,h} - \bar{p}_{c,h}| \quad (4.14)$$

Współczynnik  $r(c)$  w podstawowej wersji algorytmu jest normalizowany za pomocą podzielenia go przez ilość harmonicznych danego kandydata F0. Normalizacja ta jest rozszerzona w pracy [39, s. 5–6] tak, aby był niezależny od natężenia:

$$\tilde{r}(c) = \frac{r(c)}{H(1 - N_{0,1}(\bar{x}))} \quad (4.15)$$

Wzór liczenia współczynnika gładkości na podstawie znormalizowanego współczynnika ostrości pozostaje niezmieniony (zgodnie ze wzorem 4.11).

Wzór na istotność pojedyńczego kandydata został rozszerzony o parametr  $k$ , którego zadaniem jest zbalansowanie wkładu gładkości na ten współczynnik:

$$S(c) = l(c) * \sigma^k(c) \quad (4.16)$$

Liczenie istotności danej kombinacji jest niezmienione (zgodne ze wzorem 4.12). W danym oknie wybierana jest kombinacja kandydatów o największym współczynniku  $S$ . Po ewaluacji każdego okna analizowanego utworu wykonywane jest przetwarzanie końcowe z wykorzystaniem sąsiednich okien, które składa się z dwóch etapów: wygładzania czasowego oraz śledzenia wysokości. Wzbogacają one omawianą metodę o informację na temat kontekstu każdego z okien czasowych, pozwalając na oszacowanie spójniejszego wyniku.

Wygładzenie czasowe jest procesem przeliczenia istotności każdej z kombinacji w oparciu o sąsiednie okna czasowe. Celem tej operacji jest wybór najbardziej prawdopodobnej kombinacji na przestrzeni  $2K + 1$  okien, a nie tylko w kontekście analizowanego odcinka czasowego.

Wszystkie kombinacje poddawane są procesowi kwantyzacji z postaci częstotliwości na notację muzyczną (np. gdy kombinacja składa się z peaków  $C_1 = [261Hz, 416Hz]$  mapowane jest to na kombinację  $C'_1 = C_4, G\sharp_4$ ). Na tak przygotowanych kombinacjach liczona jest wygładzona istotność  $\tilde{S}$  z użyciem sąsiadujących okien:

$$\tilde{S}(C'_i(t)) = \sum_{j=t-K}^{t+K} S(C'_i(j)) \quad (4.17)$$

gdzie  $C'_i(t)$  to kombinacja  $i$  po procesie kwantyzacji w czasie  $t$ . W ogólności, we wszystkich  $2K$  oknach wokół okna o indeksie czasowym  $t$  wybierane są kombinacje zawierające dokładnie te same nuty a następnie sumowana jest ich istotność w celu uzyskania nowego wyznacznika, na którego podstawie wyznaczana jest najbardziej prawdopodobna kombinacja:

$$\hat{C}'(t) = \arg \max_i \left\{ \tilde{S}(C'_i(t)) \right\} \quad (4.18)$$

Tak wybierane kombinacje są bardziej spójne w domenie czasowej, niwelując gwałtowne zmiany wykrytych częstotliwości.

Śledzenia wysokości zaproponowane w pracy [39, s. 7–8] jest drugą metodą, która wymusza wybór kombinacji o gładkim w czasie przejściu z jednej na drugą. Metoda ta polega na skonstruowaniu warstwowego, ważonego, ukierunkowanego grafu acyklicznego (wDAG, z ang. *weighted directed acyclic graph*).

Graf  $G = (V, v_b, E, w, t)$  będący warstwowy wDAG składa się ze zbioru wierzchołków  $V$ , wierzchołka początkowego  $v_b$ , zbioru krawędzi  $E$ , funkcji  $w$  będącej wagą krawędzi pomiędzy dwoma wierzchołkami w sąsiednich warstwach grafu  $w(v_b, v_j)$  oraz funkcji pozycji  $t : V \rightarrow \{0, 1, 2, \dots, T\}$ . Wierzchołkami są kombinacje kandydatów F0 w danym oknie czasowym, natomiast  $t$  dla każdego wierzchołka przyporządkowuje okno czasowe, w którym rozpatrywana jest dana kombinacja. Warstwą nazywane są wszystkie wierzchołki, dla których funkcja  $t$  daje ten sam wynik  $t(v) = \tau$ . Innymi słowy, warstwą w grafie jest zbiór kombinacji w jednym oknie czasowym.

Krawędzie prowadzone są ze wszystkich wierzchołków warstwy dla  $t(v_i) = \tau$  do wszystkich krawędzi warstwy następnej  $t(v_j) = \tau + 1$ . Waga krawędzi liczona jest według wzoru:

$$w(v_i, v_j) = \frac{D(v_i, v_j)}{\tilde{S}(v_j) + 1} \quad (4.19)$$

gdzie  $\tilde{S}$  oznacza istotność kombinacji warstwy, do której prowadzona jest krawędź, liczona według wzoru 4.17 a  $D(v_i, v_j)$  jest współczynnikiem podobieństwa pomiędzy kombinacjami odpowiadającymi wierzchołkom  $v_i$  i  $v_j$  liczona jako:

$$D(v_i, v_j) = \sum_{\forall C \in v_i, v_j} |\tilde{l}(v_{i,c}) - \tilde{l}(v_{j,c})| + \sum_{\forall C \in v_i - v_j} \tilde{l}(v_{i,c}) + \sum_{\forall C \in v_j - v_i} \tilde{l}(v_{j,c}) \quad (4.20)$$

gdzie  $\tilde{l}(v_{i,c})$  oznacza ogólną głośność kandydata  $c$  w kombinacji  $i$ . Zgodnie z powyższym wzorem 4.20 koszt przejścia pomiędzy wierzchołkami, które mają więcej wspólnych kandydatów, będzie mniejsza od tych, które mają więcej rozłącznych kandydatów F0.

W trakcie budowania wDAG w celu zwiększenia wydajność wybieranych jest tylko  $M$  najbardziej prawdopodobnych kombinacji dla każdego okna czasowego. Po wygenerowaniu grafu na podstawie wag liczonych wzorem 4.19 wyznaczana jest najkrótsza ścieżka przy pomocy algorytmu Dijkstry [10] od pierwszego okna ( $t(v) = 0$ ) do ostatniego ( $t(v) = T$ ). Te wierzchołki, które należą do najkrótszej ścieżki odpowiadają kombinacjom, które są ostatecznie wybierane jako wynik algorytmu.

### 4.3. Metody z wykorzystaniem uczenia maszynowego

Jak było to opisane w sekcji 2, obecnie najbardziej efektywnymi metodami w dziedzinie transkrypcji muzyki są te, oparte na NMF oraz te, oparte na sieciach neuronowych (NN, z ang. *Neural Network*). Te drugie można zaobserwować na szczytach list wyników porównania efektywności modeli transkrypcji, takich jak MIREX (z ang. *The Music Information Retrieval Evaluation eXchange*, framework do formalnej ewaluacji wydajności algorytmów wydobywających informacje muzyczne z sygnałów cyfrowych), już w 2012 [2, s. 415–416]. Transkrypcja muzyki jest problemem ściśle powiązanym z zadaniem rozpoznawania wzorców, na którego rozwój w ostatnich latach znaczący wpływ miały metody używające NN [1, s. 26–28].

W dalszej części tej sekcji zostaną omówione dwa systemy transkrypcji oparte na sieciach neuronowych. Pierwszy z nich będący transkrypcją na poziomie notacji muzycznej

4.3.1, drugi zaś na poziomie nut 4.3.2, będący do tej pory najefektywniejszym systemem transkrypcji [1, s. 27]. W ramach tej pracy nie zostaną opisane dokładne szczegóły dotyczące działania sieci a jedynie pokrótko omówione specyficzne aspekty użytych architektur, które przyczyniły się do ich wysokich efektywności.

#### 4.3.1. Holistyczna metoda z użyciem sieci neuronowych

Praca [46] miała na celu zbudowanie architektury zdolnej do transkrypcji A2S (Audio do partytury z ang. *Audio-to-Score*). Jest to rzadkie podejście, wykorzystujące poziom abstrakcji notacji muzycznej. Wiele spośród istniejących algorytmów zakłada transkrypcję do plików MIDI lub podobnych, których struktura opiera się na zapisie rolki pianina, której poziom abstrakcji pozwala wygenerować notację muzyczną. Omawiany w tej sekcji algorytm zakłada bezpośredni proces transkrypcji cyfrowego sygnału akustycznego do notacji muzycznej.

Proces określenia partytury ma wiele różnic od tego, którego celem jest wygenerowanie transkrypcji w postaci rolki pianina. Nawet bezbłędne A2S nie jest w stanie wiernie odzwierciedlić kompozycji muzycznej z sygnału ze względu na ekspresywny charakter wykonania. Kompozycja na podstawie notacji muzycznej może być wykonana na wiele różnych sposobów, w zależności od interpretacji. Odwrotna relacja również zachodzi - to samo wykonanie ma różne reprezentacje w postaci partytury. W zależności od przyjętej szczegółowości transkrypcji, różne parametry muzyczne mogą być badane, które w dalszej kolejności przekładają się na symbole muzyczne występujące w partiturze, takie jak oznaczenia tempa, metrum (sygnatura czasowa), tonacji, w jakiej została napisana kompozycja, oznaczenia artykulacji itp.

Algorytm opisywany w [46] zakłada pewien stały alfabet notacji muzycznych, za pomocą którego można zdefiniować analizowany utwór. Opisywany A2S przedstawiony jest jako funkcja dziedziny domeny plików dźwiękowych  $f : \mathcal{X} \rightarrow \Sigma^*$ , gdzie  $\mathcal{X}$  jest domeną plików muzycznych a  $\Sigma$  jest skończonym alfabetem muzycznych symboli.

Model przyjmuje jako dane wejściowe spektralną reprezentację sygnału w czasie uzyskaną poprzez STFT z komponentami rozmieszczenymi logarytmicznie i magnitudą w skali logarytmicznej. Wynikiem algorytmu jest sekwencja symboli mogąca być zaprezentowana jako zapis nutowy. Symbole określają nuty oraz przerwy z odpowiadającym im czasem trwania, kreski taktowe, łuki pomiędzy nutami oraz fermaty.

Omawiana praca opisuje proces przygotowanych danych treningowych jako nie-

trywialny ze względu na małą ilość rzetelnych kompozycji w postaci cyfrowych notacji muzycznych. Dokładność obecnej algorytmiki systemów optycznego rozpoznawania muzyki na podstawie zapisu nutowego przez autorów uznawana jest jako niewystarczająca aby użyć jej w celu generowania danych treningowych. Ostatecznie wykorzystana została baza *humdrum-data* dostępna w repozytorium [47]. Dane te są w autorskim formacie zaproponowanym w zestawie narzędzi *humdrum*. Format ten jest przeznaczony do ogólnych celów badań nad muzyką. Syntaks *humdrum* nakreśla wzór, którego częścią są inne schematy notacji muzycznej wyższego poziomu, takie jak format  $^{**}kern$ , na którym oparte są referencyjne dane wykorzystane w opisywanym badaniu. Składnia  $^{**}kern$  jest szczególnie odpowiednia do tego modelu, ponieważ zaprojektowana jest ona do kodowania alfabetu muzycznego muzyki zachodniej.

Zadanie A2S zadeklarowane jest jako pozyskanie najprawdopodobniejszej sekwencji symboli muzycznych występującej w kompozycji w pliku audio  $x \in \mathcal{X}$ :

$$\hat{s} = \arg \max_{s \in \Sigma^*} P(s|x) \quad (4.21)$$

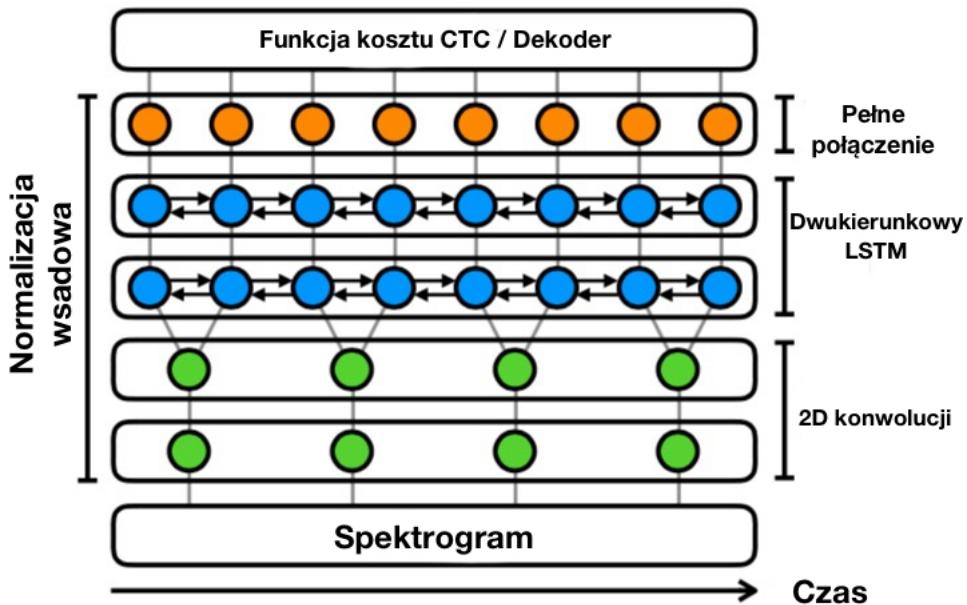
Podejście holistyczne do rozwiązania zakłada użycie konwolucyjnej rekurencyjnej sieci neuronowej (CRNN, z ang. *Convolutional Recurrent Neural Network*). Architektura CRNN składa się z bloku warstw konwolucyjnych oraz bloku warstw rekurencyjnych. Ten pierwszy odpowiedzialny jest za wydobywanie cech z sygnału wejściowego, drugi zaś odpowiada za wykrywanie cech w kontekście sekwencyjnym.

Przy  $W$  będącym długością sygnału wejściowego (w ilości okien czasowych) długość wektora cech będzie równa  $L = \gamma W$ , gdzie  $\gamma \leq 1$ , co jest długością aktywacji ostatniej warstwy bloku konwolucyjnego. Wynik ostatniej warstwy konwolucyjnej jest używany jako wejście do bloku warstw rekurencyjnych, którego aktywacja ostatniej warstwy jest estymacją danego okna:

$$P(\sigma|x, j), 1 \leq l \leq L, \sigma \in \Sigma \quad (4.22)$$

Dekodowanie odbywa się poprzez wybranie najbardziej prawdopodobnego symbolu dla każdego wektora cech danego bloku rekurencyjnego, a następnie wyniki są ze sobą łączone i usuwane są symbole puste [46, s. 2–5] .

Architektura zaprezentowana w omawianej pracy przedstawiona jest na rysunku 4.10. Pierwsze dwie warstwy konwolucyjne nakładają 16 filtrów o wymiarach 3 x 3 z krokiem równym 2 na osi częstotliwości. Kolejne dwie warstwy są blokiem rekurencyjnym,



Rysunek 4.10: Architektura sieci neuronowej zastosowanej w pracy [46]. Diagram z [46, s. 5], tłumaczenie własne.

w którego skład wchodzą dwukierunkowe sieci z długotrwałą pamięcią krótkoterminową (LSTM, z ang. *Long Short-Term Memory*), każda z nich z 1024 ukrytymi jednostkami. W celu uniknięcia nadmiernego dopasowania dodane są sieci normalizacji wsadowej (ang. *Batch Normalization*) pomiędzy każdą warstwą z wyjątkiem pierwszej i ostatniej, warstwy *dropout* po każdej warstwie konwolucyjnej oraz ostatniej warstwie rekurencyjnej, które przepuszczają tylko fragment danych. Szczegółowy opis użytych parametrów w zależności od danych testowych jest zawarty w omawianej pracy [46].

#### 4.3.2. Onsets and Frames

Użycie sieci neuronowych do modelowania systemu transkrypcji występuje w literaturze naukowej już od dłuższego czasu. Poprawność wyników tych architektur nie była znaczowo większa od równolegle wynajdywanych algorytmów korzystających z innych narzędzi niż NN. W ostatnich latach, w ramach projektu Magenta [11] powstał system transkrypcji o nazwie *Onsets And Frames* (początki dźwięków i okna, tłumaczenie własne), który, według ewaluacji przedstawionej w dedykowanej pracy naukowej [17], osiąga rekordowe rezultaty w poprawności wyników.

Projekt Magenta ma na celu wytworzenie algorytmiki i oprogramowania, które ma być narzędziem w rękach artystów. Ma to na celu udowodnienie, że uczenie maszynowe

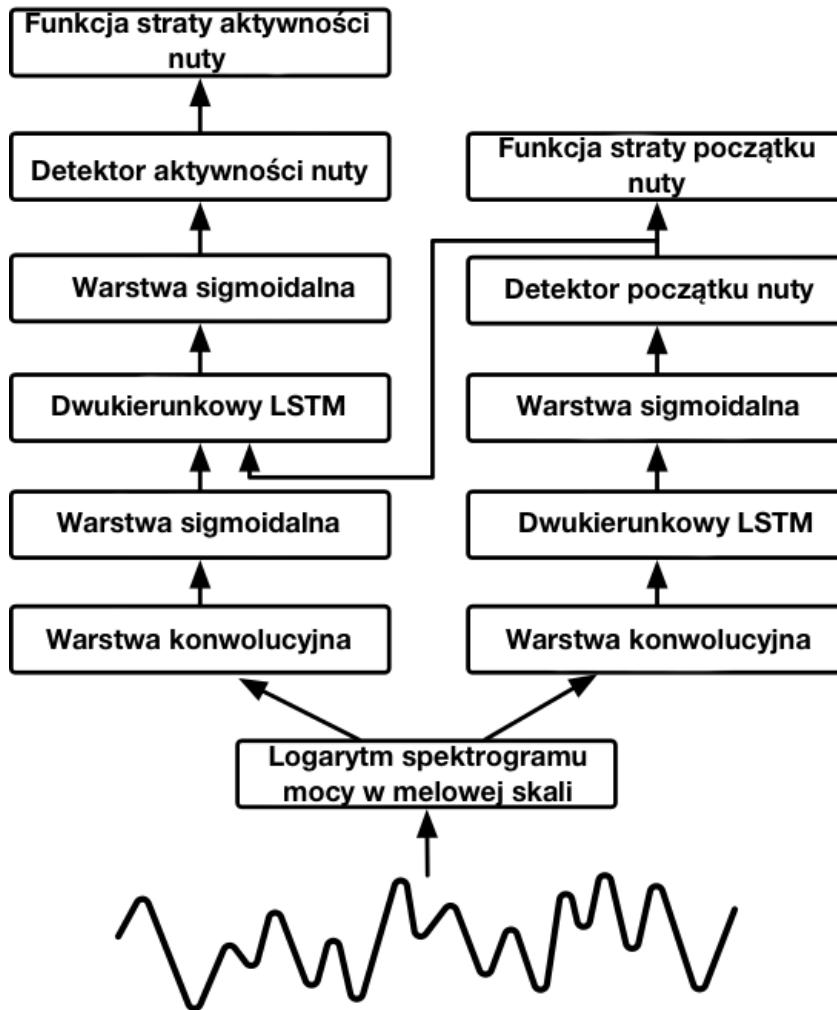
może być pomocne w procesie artystycznie twórczym. Rozwiązania te są rozpowszechnione jako otwarte oprogramowanie, ułatwiając tym samym do nich dostęp. Douglas Eck, członek zespołu pracującego nad Magenta zapewnia, że projekt ten ma na celu dać źródło inspiracji artystom, a nie szukać sposobów ich zastąpienia przez komputery. System transkrypcji, który został opracowany w ramach tego projektu, jest nie tylko docelowym produktem tej koncepcji, ale też narzędziem pozwalającym ewaluować i zasilać danymi inne badane przez Magenta architektury.

Głównym założeniem, które odróżnia Onsets and Frames od innych systemów transkrypcji bazujących na sieciach neuronowych, jest rozłożenie zadania na dwa problemy: wykrywanie początku nuty oraz wykrywanie aktywności nut w oknie. Oba podzadania mają osobne dedykowane sieci. Nuta jest uznawana za obecną wtedy, i tylko wtedy, gdy zostanie dla niej wykryty początek oraz aktywność.

Dane wejściowe są w postaci spektrogramów w skali melowej i logarytmie amplitudy, z 229 logarytmicznie rozmieszczonymi w domenie częstotliwości komponentami, oknem długości 2048 sampli i odstępami pomiędzy oknami długości 2048. Dane równolegle zasilają obie sieci. Wynik detektora początków nut jest dostarczany do sieci aktywacji nuty, dając jej łącznie dwie informacje wejściowe - czy początek nuty został wykryty oraz spektrogram sygnału.

Detektor początku nuty składa się z modelu akustycznego (jego architektura opisana jest w [24, s. 1–3]) połączonego z dwukierunkowym LSTM o 128 jednostkach dla każdego kierunku, który połączony jest z warstwą sigmoidalną, która ma 88 wyjść. Liczba wyników jest równa ilości klawiszy na pełnowymiarowym pianinie. Wykrywacz aktywności nuty zbudowany jest z modelu akustycznego połączonego z sigmoidalną warstwą, analogiczną do tej w detektorze. Jej wynik jest łączony z wynikiem detektora początku nuty jako wejście do dwukierunkowego LSTM z 128 jednostkami dla każdego kierunku. Po tym występuje kolejna warstwa sigmoidalna z analogicznymi do poprzednich parametrami, której wynik, po nałożeniu progu 0.5, jest interpretowany jako predykcja danego okna. Architektura ta jest przedstawiona na rysunku 4.11

Metoda Onsets and Frames wprowadziła detekcje dynamiki, która nie została rozpatrzona przez żaden z istniejących wcześniej modeli. Model zbudowany jest analogicznie, ale nie jest połączony z architekturą opisaną powyżej. Etykiety dynamiki są wyliczane przez iloraz każdej dynamiki do maksymalnej dynamiki obecnej w danym fragmencie. Naj-



Rysunek 4.11: Diagram przedstawiający architekturę modelu *Onesets and Frames*. Diagram z [16], tłumaczenie własne.

mniejsza dynamika jest wyznaczana jako  $\frac{v_{min}}{v_{max}}$ . Wynik jest ograniczany do zakresu  $[0, 1]$  a następnie przekształcany do parametru *velocity* w standardzie MIDI funkcją mapującą, która ma postać:

$$v_{midi} = 80v_{predicted} + 10 \quad (4.23)$$

gdzie  $v_{predicted}$  jest wyliczoną etykietą dynamiki z zakresu  $[0, 1]$ .

Opisywana praca została zaktualizowana, co zostało opisane w [20, s. 3–5]. Pierwotnie, model trenowany i ewaluowany był przy użyciu bazy MAPS [12]. Na czas powstawania pierwszej wersji rozwiązania była ona najbardziej odpowiednia, biorąc pod uwagę ilość i jakość zawartych w niej danych. W nowszej wersji została ona zastąpiona bazą MAESTRO (z ang. *MIDI and Audio Edited for Synchronous TRacks and Organization*) przygotowaną w ramach opisywanej pracy. MAESTRO zawiera ponad 172.3 godziny spa-

rowanych plików audio z MIDI, co jest trzykrotnie większą ilością danych od bazy MAPS. Zawarte pliki audio są nagraniami wystąpieniami pianisty na fortepianie akustycznym Yamaha Disklavier. Zdarzenia nut w pliku MIDI nakładają się z tymi występującymi w plikach audio z dokładnością do około 3 milisekund. Dzielenie danych na treningowe, walidacyjne i testowe jest wykonane w taki sposób, że ta sama kompozycja, nawet jeżeli wykonana przez różnych artystów, nie występuje w więcej niż jednej grupie. Zasada ta wprowadzona jest w celu uniknięcia efektu nadmiernego dopasowania (z ang. *overfittingu*) [18].

Poza użyciem innej bazy, zmiany do Onsets and Frames zaproponowane w [20, s. 3–5] zakładają dodanie detekcji offsetu, która zasila detektor aktywności nut, choć nie jest bezpośrednio używana w dekodowaniu. Zaproponowana jest również zmiana dwukierunkowych warstw LSTM z 128 do 256 jednostek, zmieniając ilość filtrów w warstwie konwolucyjnej z 32/32/64 na 48/48/96, zwiększać tym samym warstwy sigmoidalne. Zmodyfikowane zostały również przejścia pomiędzy podsieciemi. Argumentowane są tym, że dla większych baz efektywność rośnie przy większym i prostszym modelu.

Dodatkowo zaproponowana została metoda augmentacji sygnału wejściowego, która polegać ma na modyfikacji każdego z danych wejściowych o pewną modulację audio (np. przesunięcie częstotliwości, dodanie szumu, zastosowanie equalizera). Zmiana ta pozytywnie wpływa na jakość modelu trenowanego bazą MAPS, jednak w przypadku bazy MAESTRO odnotowany został minimalny spadek w dokładności wyniku.

## 5. Implementacja

W tym rozdziale opisane są implementacje algorytmów transkrypcji z rozdziałów 3 i 4, które są częścią projektu powstałego w ramach tej pracy magisterskiej. Kod napisany został w skryptowym języku Python w wersji 3.7.4, który wydawał się najodpowiedniejszy do tego typu zadania ze względu na dostępne biblioteki dedykowane do matematycznych problemów (jak *numpy* czy *scipy*). Szczegółowa lista użytych rozwiązań dostępna jest w pliku README.md dostępnym w głównym katalogu projektu.

Wszystkie z funkcji operujących na danych audio opisanych poniżej zakładają pliki o rozszerzeniu *.wav* z kodowaniem PCM o 16 lub 32 bitach na sampel. Sygnały, które są dwukanałowe (stereo audio) zostają przed analizą „spłaszczone” do jednego kanału poprzez przemnożenie wartości każdego z nich przez  $\frac{1}{2}$  oraz scalenie ich razem poprzez sumę wartości tych wektorów. Dane przed analizą są znormalizowane poprzez skalowanie danych do zakresu  $[0, 1]$ .

Każda z funkcji implementujących algorytmy transkrypcji muzyki przyjmuje cztery bazowe argumenty odpowiadające danym wejściowym i sposobie ich fragmentacji na okna czasowe:

- *data* - znormalizowane dane pobrane z pliku dźwiękowego,
- *frameWidth* - długość okna w samplach,
- *sampleRate* - częstotliwość próbkowania,
- *spacing* - odstęp pomiędzy kolejnymi oknami danych.

```

1  for i in range(0, int(math.ceil((len(data) - frameWidth) / spacing))):
2      frame = data[i*spacing:i*spacing+frameWidth] * hann
3      frame = np.concatenate((frame, zeroPadding))
4      frameComplex = fft(frame)

```

*Listing 5.1: Przykład pętli po danych wejściowych z uwzględnieniem okna czasowego. Takie dzielenie danych, z nakładaniem funkcji okna i dodaniem marginesu zer jest wykonywane w większości zaimplementowanych algorytmów.*

Sposób iterowania pomiędzy kolejnymi oknami jest również identyczny dla każdej z implementacji i jest przedstawiony na listingu 5.1. W lini 2 listingu na konkretne okno danych nakładana jest funkcja okna (w tym przypadku okno Hanna). W przypadku algorytmów operujących na spektrum, do okna dodawana jest odpowiednia ilość zer (linia 3)

jako margines zer, i dopiero na tak przygotowanych danych wykonywana jest transformata (linia 4).

Zaimplementowany został również prosty mechanizm porównujący pliki MIDI. Jego celem jest porównanie wyników zaimplementowanych algorytmów, co zostało opisane w sekcji 5.8. Wyniki ewaluacji przeprowadzonej na zaproponowanej implementacji opisane są następnie w rozdziale 6. Na potrzeby tej pracy przedstawienie wyników w postaci MIDI zostało zaimplementowane dla algorytmów wykrywających F0 w sygnale monofonicznym i polifonicznym, czego opis można znaleźć w sekcji 5.7.

Poza algorytmiką w ramach tej pracy magisterskiej zaimplementowany został również graficzny interfejs (GUI, z ang. *Graphical User Interface*) w postaci aplikacji webowej z dedykowanym serwerem lokalnym napisanym w Python, za pomocą którego można łatwo przetestować działanie zaimplementowanych metod. Krótki opis tego rozwiązania znajduje się w sekcji 5.9.

### 5.1. Autokorelacja

Funkcja autokorelacji (opisana w rozdziale 3.1) jest jedynym z zaimplementowanych algorytmów, który operuje na nieprzetworzonym sygnale w domenie czasu. Dodatkowymi parametrami (poza tymi związanymi z parametrami danych i długością okna) są częstotliwości maksymalna i minimalna (co jest pokazane na listingu 5.2), jakie są w zakresie analizy. Na ich podstawie wyznaczane są minimalne i maksymalne przesunięcie, które w dalszej części ograniczają zakres, na którym operuje funkcja autokorelacji (co jest pokazane na listingu 5.3).

```
1 minLag = int(floor(sampleRate / fqMax))
2 maxLag = int(ceil(sampleRate / fqMin))
```

*Listing 5.2: Wyliczanie maksymalnego i minimalnego przesunięcia funkcji autokorelacji*

```
1 n = len(data)
2 result = list(np.zeros(min_lag))
3 for lag in range(min_lag, max_lag):
4     sumarray = np.zeros(n+lag)
5     sumarray[:n] = data
6     sumarray[:n-lag] *= data[lag:]
7     sum = np.sum(sumarray[:n-lag])
8     result.append(float(sum/(n-lag)))
9 return result
```

Listing 5.3: Funkcja autokorelacji

Funkcja autokorelacji wyliczana jest dla każdego okna czasowego. Zwraca ona wektor długości  $\minLag - \maxLag$  wartości korelacji danych wejściowych z danymi przesuniętymi o  $lag$  sampli (próbka o indeksie  $i$  po przesunięciu ma indeks  $i + lag$ ). Dla danego okna jako estymację F0 wybierane jest to przesunięcie, które ma najwyższy współczynnik korelacji. Implementacja znajduje się w projekcie w pliku *server/transcription/ac.py*.

## 5.2. Cepstrum

Cepstrum (opisane w rozdziale 3.3) w projekcie używany jest w dwóch celach - jako trzon metody do estymacji F0 jak i do filtrowania sygnału w domenie quefrencji. Z tego powodu implementacja funkcjonalności została wyniesiona do pomocniczego pliku *utils/cepstrumUtils*. Funkcja *real\_cepst\_from\_signal* przyjmuje okno danych, na którym wyliczane jest cepstrum mocy według wzoru 3.5 co przedstawione jest na listingu 5.4.

```

1  spectrum = fft(data)
2  logSp = np.log(np.abs(spectrum))
3  ceps = np.abs(ifft(logSp)) ** 2
4  return ceps, logSp, spectrum

```

Listing 5.4: Wyliczanie cepstrum mocy

Estymacja F0 dla każdego z okien czasowych polega na transformacji danych do cepstrum oraz znalezienie komponentu cepstralnego o największej wartości. Podzielenie quefrencji wyznaczonego komponentu przez częstotliwość próbkowania daje w wyniku częstotliwość F0. Pod uwagę branych jest jedynie  $n/2$  pierwszych elementów cepstrum, gdzie  $n$  jest ilością danych wejściowych łącznie z marginesem zer (ze względu na symetryczny charakter wyniku cepstrum). Pierwsze elementy wyniku cepstrum nie sąbrane pod uwagę przy szukaniu maksymalnego komponentu ze względu na znaczne zwiększenie błędności wyniku (pierwsze komponenty cepstralne symbolizują podobieństwo sygnału do samego siebie, analogicznie do autokorelacji), co zostało w implementacji zrobione poprzez wyzerowanie wartości pierwszych 14 elementów wektora cepstrum (linia 4 listingu 5.5).

```

1  cepst, logSp, spectr = real_cepst_from_signal(frame)
2  fftLen = int(np.floor(len(spectr)/2))
3  cepst = cepst[:fftLen//2]
4  cepst[0:14] = np.zeros(14)

```

```
5 cepstra.append(cepst)
6 maxperiod = np.argmax(cepst)
7 bestFq.append(sampleRate/maxperiod)
```

*Listing 5.5: Estymacja F0 przy pomocy cepstrum*

W ramach tej pracy magisterskiej została zaimplementowana wersja analizy cepstralnej przy użyciu karty graficznej GPU (karty graficznej, z ang. *graphics processing unit*). Zaimplementowane rozwiązanie zakłada dostęp do API CUDA, z którego korzysta użyta biblioteka dająca dostęp do wywoływania poleceń GPU z poziomu kodu napisanego w języku Python. Szczegóły karty jak i stopień uzyskanego przyspieszenia opisane są w rozdziale 6.

Rozwiązanie GPU znajduje się w pliku *server/utils/cepsUtilsGpu.py* w postaci klasy *CepsUtilsGpu* implementującej interfejs biblioteki *reikna* pozwalający na tworzenie planu obliczeń na GPU. Plan ten przyjmuje cztery argumenty:

- znormalizowane dane audio,
- długość okna,
- długość marginesu zer,
- odległość pomiędzy oknami.

Dane poprzez transformację dostosowywane są do formatu wejścia funkcji FFT dostępnej w bibliotece *reikna*. Cały proces, przedstawiony na listingu 5.6, można opisać w następujący sposób:

1. dane sygnału wejściowego (*flat\_input*) przekształcane są do tablicy dwuwymiarowej *unwindowed\_input* transformacją *rolling\_frame* tak, że długości wierszy są równe długości ustalonego okna czasowego z długością marginesu zer. Dane składowane są po kolej z uwzględnieniem marginesu zer i odległości pomiędzy oknami,
2. tablica *unwindowed\_input* przekształcana jest do *input\_real* transformacją *hanning\_window*, która nakłada funkcję Hanna do każdego z wierszy danych wejściowych,
3. dlatego, że do użytej implementacji FFT wymagane jest podanie danych o typie zespolonym, użyta została transformacja *combine\_complex*, która przyjmuje dwa parametry - dane rzeczywiste, którymi są przygotowana tablica *input\_real* oraz dane urojone,

```

1  fft = FFT(fft_arr, axes=(1,))
2  fft.parameter.input.connect(
3      to_complex_trf, to_complex_trf.output,
4      input_real=to_complex_trf.real, input_imag=to_complex_trf.imag)
5  fft.parameter.input_imag.connect(broadcast_zero_trf, broadcast_zero_trf.output)
6  fft.parameter.input_real.connect(window_trf, window_trf.output,
7      unwindowed_input=window_trf.input)
8  fft.parameter.unwindowed_input.connect(
9      rolling_frame_trf, rolling_frame_trf.output, flat_input=rolling_frame_trf.input)
10 fft.parameter.output.connect(
11     log_pow_trf, log_pow_trf.input, log_pow=log_pow_trf.output)
12 ifft = FFT(Type(complex_dtype, fft.parameter.log_pow.shape), axes=(1,))
13 ifft.parameter.input.connect(
14     to_complex_trf, to_complex_trf.output,
15     input_real=to_complex_trf.real, input_imag=to_complex_trf.imag)
16 ifft.parameter.input_imag.connect(broadcast_zero_trf, broadcast_zero_trf.output)
17 crop_trf = crop_frequencies_ceps(ifft.parameter.output)
18 ifft.parameter.output.connect(
19     crop_trf, crop_trf.input, cropped_amplitude=crop_trf.output)

```

*Listing 5.6: Proces wyliczania cepstrum przy użyciu GPU*

które są wygenerowane za pomocą funkcji *broadcast\_const* i mają postać tablicy zer takich samych wymiarów jak *input\_real*,

4. na tak przygotowanych danych wykonywana jest funkcja FFT, po której stosowana jest transformacja *log\_pow* dająca w wyniku *log\_pow* będące tablicą spektrum logarytmów mocy,
5. funkcja IFFT (odwrotność FFT) wywoływana jest na wyniku kolejnej transformacji *combine\_complex*, której danymi wejściowymi rzeczywistymi jest tablica *log\_pow* a urojonymi jest tablica generowana analogicznie do tej, przygotowanej w pkt. 3,
6. wynik IFFT przekształcany jest transformacją *crop\_frequencies\_ceps*, która zeruje pierwszych 10 wartości (analogicznie do 5.5) oraz wycina elementy przekraczające częstotliwość Nyquista,
7. wynik *crop\_frequencies\_ceps* jest transponowany i zwracany jako wynik.

### 5.3. ACLOS

Trzonem metody ACLOS (opisanej w 3.4) jest funkcja autokorelacji w domenie częstotliwości, której implementacja jest bardzo podobna do tej, w domenie czasu, opisanej

```

1   k = np.arange(len(ceps))
2   h = (coefficient / 2.)
3   def f(lifterType):
4       lift = {
5           LifterType.sine: 1 + h * np.sin(np.pi*k/coefficient),
6           LifterType.triangle: 1 + h * (k - 1) / (coefficient - 1),
7           LifterType.rectangle: np.ones(len(ceps))
8       }[lifterType]
9       return lift
10
11 lift = f(lifterType)
12 return ceps * lift

```

Listing 5.7: Filtrowanie w domenie quefrencji

```

1 interp_x = np.linspace(0, dataLen-1, num=dataLen*interpolMultiplier)
2 argmax = np.argmax(interpolatedAutocorrelation(interp_x))
3 correlationArgMax = int(np.round(argmax / interpolMultiplier))
4 delta = argmax - (correlationArgMax * interpolMultiplier)
5 bestFq = fftToFq[correlationArgMax] + (fqMaxError * delta / interpolMultiplier)
6 return bestFq

```

Listing 5.8: Wyznaczanie F0 w ACLOS

w sekcji 5.1. Wyróżniającymi się cechami tego algorytmu są użycie lifteringu na sygnale oraz wykorzystanie interpolacji do zwiększenia dokładności wyniku.

Filtrowanie w domenie quefrencji odbywa się dla każdego okna czasowego, po przekształceniu i nałożeniu funkcji okna oraz przekształceniu do postaci spektrum mocy. Widmo przekształcane jest do cepstrum przy pomocy IFFT, po czym wykonywane jest na nim filtrowanie. Proces ten przedstawiony jest na listingu 5.7. Poza danymi wejściowymi jako argumenty funkcja *lifterOnCeps* przyjmuje typ lifteringu *lifterType* (którym może być funkcja sinusoidalna, trójkątna lub prostokątna) oraz mnożnik *coefficient*.

Estymacja F0 przyjmuje interpolację sześcienną wyniku funkcji autokorelacji. Na jej podstawie wyznaczana jest częstotliwość z najlepszym wynikiem, co przedstawione jest na listingu 5.8. Zmienna *fftToFq* jest tablicą z wartościami równymi częstotliwości dla kolejnych indeksów wyniku FFT dla danej rozdzielczości. Maksymalny błąd w domenie częstotliwości wiąże się z rozdzielczością wykonywanego FFT i jest równy  $\frac{f_s}{NFFT}$ , gdzie  $f_s$  jest częstotliwością próbkowania, a  $NFFT$  to ustalona długość okna.

## 5.4. Metoda Pertusa i Iñesta (2008)

Metody opisane w sekcjach 4.2.4 oraz 4.2.5 w dużej części się pokrywają, dlatego też implementacja obu metod znajduje się w tej samej funkcji *harmonicAndSmoothness-BasedTranscription* w pliku *server/transcription/generativeMethodByPertusAndInesta.py*. W przypadku, gdy argument *newAlgorithmVersion* przyjmuje wartość *False*, zostanie wykonana metoda zgodna z opisem pracy z 2008 roku [40], którego implementacja opisana jest w tej sekcji. W przeciwnym wypadku zostanie wykonana znowelizowana wersja algorytmu z 2012 roku ([39]). Trzon metody zaprezentowany jest na listingu 5.9.

W pierwszej kolejności wywoływana jest funkcja *coreMethod*, której zadaniem jest przetworzenie sygnału do postaci nut, na którym następnie wykonywane jest przetwarzanie końcowe. Z każdego spektrum okna czasowego po nałożeniu filteringu wydobywane są wartości (piki) o amplitudach powyżej zadanego progu i na ich podstawie wyznaczani są kandydaci do F0. Zbiór wyznaczonych częstotliwości pomniejszany jest o te, które nie mają wystarczającej ilości harmonicznych w tablicy pików. Liczba kandydatów następnie jest posortowana względem amplitud i odrzucone zostają wszystkie przekraczające indeks *maxCandidates* zmniejszając tym samym liczbę kombinacji w przypadku dużej liczby kandydatów.

Kolejnym krokiem jest wyznaczenie wszystkich możliwych kombinacji i wyliczenie istotności każdej z nich. Funkcja obliczająca istotność tworzy kopie tablicy pików i tablicy wszystkich wzorów spektralnych, a następnie iteruje po wszystkich kandydatach danej kombinacji i wykonuje dwa kroki:

1. kopia pików i wzorów spektralnych zostają wygładzone w oparciu o spektrum danego kandydata funkcją *smoothenPattern* zgodnie z metodą opisaną w rozdziale 4.2.4;
2. wyliczane zostają głośność jako suma amplitud oraz gładkość danego wzoru spektralnego kandydata F0.

W danym oknie czasowym wybierana jest kombinacja o największej sumie istotności harmonicznych. Lista częstotliwości kandydatów F0 wygranej kombinacji transformowana jest na tablicę, której indeksy oznaczają numery nut MIDI a wartościami są parametry dynamiki, wyliczane jako średnia amplituda F0 wraz z harmonicznymi.

Przetwarzanie końcowe zaimplementowane jest w funkcji *postProcessMidiNotes* i wylicza ono dynamikę na podstawie wcześniej nadanej wartości z amplitudy i generuje

bardziej „spójną” rolkę pianina, odrzucając zbyt krótkie nuty i łącząc w całość nuty, które były wykryte w oknie o indeksie  $n - 1$  oraz  $n + 1$ , ale nie były wykryte w oknie o indeksie  $n$ . Implementacja pozwala na zdefiniowanie ilości sąsiadujących nut branych pod uwagę podczas tego procesu poprzez parametr *neighbourMerging*.

```

peaks, candidate = getPeaksAndCandidates(countPowerFftWindow(i))
hypotheses, amplitudeSum, patterns, ownerships =
    getCandidatesThatHaveEnoughHarmonics(candidate, peaks)
if len(hypotheses) == 0:
    resNotes.append({})
    continue
sortedByFq = getMaxCandidatesByPower(amplitudeSum, hypotheses)
possibleCombinations = []
for num_pitches in range(1, min(maxParallelNotes, len(sortedByFq)) + 1):
    for combo in combinations(sortedByFq, num_pitches):
        possibleCombinations.append(combo)
allSaliences = []
allMidiNotes = []
for combination in possibleCombinations:
    combinationSalience = countCombinationSalience(
        combination, patterns, peaks, ownerships)
    allSaliences.append(combinationSalience)
    midiNotes = {}
    for fftFq in combination:
        fq = fft_to_hz_array[fftFq]
        midiNotes[hz_to_midi(fq)] = (amplitudeSum[fftFq] / len(patterns[fftFq]))
    allMidiNotes.append(midiNotes)
resNotes.append(allMidiNotes[np.argmax(allSaliences)])

```

*Listing 5.9: Trzon algorytmu Pertusa i Iñesta (2008)*

## 5.5. Metoda Pertusa i Iñesta (2012)

Gdy funkcja *harmonicAndSmoothnessBasedTranscription* zostanie wywołana z wartością *True* argumentu *newAlgorithmVersion* wykonana zostanie metoda z usprawdzeniami opisanymi w sekcji 4.2.5 i pracy naukowej [39]. Implementacja różni się od tej opisanej w 5.4 sposobem wyliczania istotności kombinacji oraz przetwarzaniem końcowym.

Proces wyliczania istotności dla danego kandydata F0 zgodnie z opisem z [39] jest przedstawiony na listingu 5.10. Wynikiem trzonu algorytmu, w przeciwieństwie do wersji z 2008 roku, jest lista nut MIDI dla wszystkich kombinacji razem z ich istotnością. Na tej podstawie w funkcji *flattenCombination* wykonywane jest wygładzanie na przestrzeni rolki pianina, jak jest to opisane w algorytmie, natomiast śledzenie wysokości

nut zaimplementowane jest w funkcji *pitchTracking*. Funkcja wyliczającą wagi każdego z wierzchołków grafu przedstawiona jest na listingu 5.11, gdzie argumenty *lvi* i *lvj* są wzorami spektralnymi harmonicznej z okna *i* i *j = i + 1* a argument *saliences* jest istotnością *j*-ego okna. Wyznaczona tablica nut jest przetwarzana metodą *postProcessMidiNotes* tak samo jak w implementacji 5.4.

```

currPattern, currPeaks = smoothenPattern(currPatterns[fundamental],
    currPeaks, ownerships)
harmonicsPatterns.append((hz_to_midi(fft_to_hz_array[fundamental]), currPattern))
currPatternPowers = np.array(currPattern)
currPatternPowers = currPatternPowers.T[1]
totalPatternLoudness = np.sum(currPatternPowers)
highestLoudness, lowestLoudness = updateMinMaxL(totalPatternLoudness,
    highestLoudness, lowestLoudness)
if lowestLoudness < highestLoudness * gamma:
    return 0.0, harmonicsPatterns
normalizedHpsPowers = np.array(currPatternPowers) / np.max(currPatternPowers)
if len(currPatternPowers) > 2:
    lowPassedConv = np.convolve(normalizedHpsPowers, gaussianPoints, 'same')
    currPatternSharpnessMeasure = np.sum(abs(lowPassedConv - normalizedHpsPowers)) /
        (len(currPatternPowers) - len(currPatternPowers) * gaussianPoints[1])
else:
    currPatternSharpnessMeasure = 0
currPatternSmoothness = 1 - currPatternSharpnessMeasure
combinationSalience += (totalPatternLoudness *
    currPatternSmoothness ** smoothnessImportance) ** 2

```

Listing 5.10: Wyliczanie istotności w algorytmie Pertusa i Iñesta (2012)

```

def countWeight(lvi, lvj, salience_j):
    D = 0
    fundamentals_i = map(lambda x: x[0], lvi)
    fundamentals_j = map(lambda x: x[0], lvj)
    for pattern_i in lvi:
        if pattern_i[0] in fundamentals_j:
            D += np.abs(sum(map(lambda a: a[1], pattern_i[1])) + sum(map(lambda a: a[1],
                [item for item in lvj if item[0] == pattern_i[0][0][1]])))
        else:
            D += sum(map(lambda a: a[1], pattern_i[1]))
    for pattern_j in lvj:
        if pattern_j[0] not in fundamentals_i:
            D += sum(map(lambda a: a[1], pattern_j[1]))
    return D / (salience_j + 1)

```

Listing 5.11: Funkcja wyliczająca wagę krawędzi w algorytmie śledzenia wysokości nut

## 5.6. Onsets and Frames

Implementacja wykorzystania metodologii *Onsets and Frames* w projekcie wykonana została na zwór tej, zaproponowanej przez autorów [17]. Udostępnili oni przykładowy kod, przy pomocy którego można przetestować działanie wytrenowanego bazą MAESTRO modelu w interaktywnym arkuszu Google Colab [36]. Wprowadzone modyfikacje miały na celu zintegrowanie tego rozwiązania do metod ewaluacyjnych opisywanego projektu.

Metody związane z inicjalizacją modelu i transkrypcją utworu zostały umieszczone w klasie *OnsetsAndFramesImpl* znajdującej się w pliku *transcription/onsetsAndFrames.py*. Instancja tej klasy, przed wywołaniem metody przeznaczonej do transkrypcji, musi zostać zainicjalizowana przy pomocy metody *initializeModel*. Model zostaje zdefiniowany i skonfigurowany na podstawie już wytrenowanego modelu, który szukany jest w katalogu *transcription/train*. Model można pobrać z repozytorium projektu *Onsets and Frames* [45]. Obiekt przechowuje informację o tym, czy już został zainicjalizowany ignorując kolejne wywołania metody *initializeModel*.

Metoda transkrypcji *transcribe* przyjmuje dwa argumenty:

- *uploaded* - plik audio do transkrypcji w postaci słownika, w którym klucz jest nazwą kompozycji a wartość danymi z pliku.
- *responseFilePath* - ścieżka do pliku, w którym ma zostać zapisana transkrypcja w postaci MIDI.

## 5.7. Generowanie MIDI

Zaimplementowane algorytmy estymujące F0, które operują na sygnale monofonicznym, zwracają wynik w postaci tablicy wyznaczonych częstotliwości dla każdego okna czasowego. W celu wygenerowania MIDI z wyniku o takiej postaci, tworzona jest tablica dwuwymiarowa  $n \times 127$ , gdzie n jest długością przekształcanej tablicy wyniku algorytmu. Każdy z wierszy reprezentuje rolkę pianina w danym oknie czasowym o indeksie równym indeksowi wiersza. Wartości tych tablic są liczbami z zakresu  $[0, 127]$  oznaczającymi dynamikę danej nuty (parametr *velocity* standardu MIDI), gdzie 0 oznacza, że dana nuta nie została wykryta. Opisywane algorytmy nie mają mechanizmu wyznaczania dynamiki wykrytych F0, dlatego też została przyjęta stała wartość *velocity* = 100 dla wszystkich wyznaczonych F0. Implementacje tego mechanizmu przedstawia listing 5.12.

```

resultPianoRoll = []
for fq in resInFOPerFrame:
    pianoRollRow = np.zeros(127)
    pianoRollRow[hz_to_midi(fq)] = 100
    resultPianoRoll.append(pianoRollRow)

```

*Listing 5.12: Funkcja wyliczająca wagę krawędzi w algorytmie śledzenia wysokości nut*

Tak przygotowana tablica rolek pianina przetwarzana jest przy pomocy funkcji *post\_process\_midi\_notes*, której zadaniem jest utworzenie listy wydarzeń MIDI na podstawie sąsiednich nut. Funkcja ta używana jest również przez algorytmy 5.4 i 5.5 jako przetwarzanie końcowe, które sterowane jest przy pomocy argumentów odpowiadających za minimalną długość nuty (*minNoteMs*), minimalną dynamikę nuty (*minNoteVelocity*) oraz maksymalną dozwoloną „lukę” w ciągłości wykrywania danej nuty (*neighbourMerging*).

Wydarzenia MIDI składają się z czterech parametrów:

- wysokości nuty (*pitch*),
- dynamiki (*velocity*),
- czasu początku nuty w ms. (*onsetS*),
- czasu trwania nuty w ms. (*durationS*).

Użyta biblioteka do generowania plików MIDI (*midiutil*) pozwala na dodanie dwóch dodatkowych parametrów do wydarzenia nuty MIDI - numer śladu (z ang. *track*), na którym znajduje się zdarzenie oraz numer kanału (z ang. *channel*) z przedziału [0, 1]. Z uwagi na charakter algorytmów, oba parametry ustawiane są na wartość 0 dla każdej nuty. Ze względu na to, że zdarzenia w czasie opisywane są w sposób absolutny (w ms.) a nie relatywny (w notacji muzycznej), parametr tempa MIDI jest nieistotny.

## 5.8. Porównywanie MIDI

Ocena wyników transkrypcji w projekcie odbywa się na wzór ewaluacji opisanych w pracach [40, s. 107] i [39, s. 8], gdzie wykorzystane były miary jakości F1 i dokładności. Do porównania dwóch plików MIDI zaimplementowana została funkcja *compare\_midi\_to\_ground\_truth*. Na potrzeby ewaluacji wyników transkrypcji została

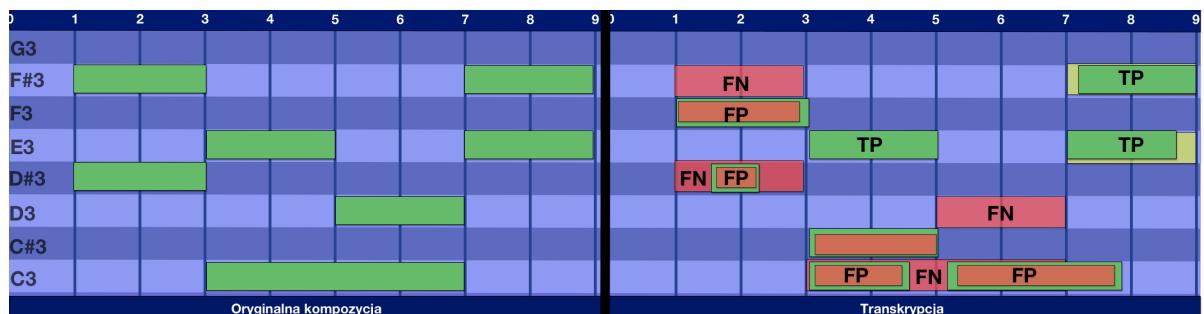
napisana funkcja `load_midi_file` pozwalająca wczytać istniejący plik MIDI w przyjętej strukturze. Obie te funkcje znajdują się w pliku `server/utils/midi.py`.

Z listy zdarzeń MIDI wczytanego pliku brane pod uwagę są tylko trzy typy: `set_tempo`, informujące o tempie wczytanej kompozycji, `note_on`, które przechowuje czas początku nuty oraz `note_off`, które informuje o końcu danej nuty. Z wyżej wymienionych zdarzeń dwa ostatnie przechowują również informacje o wysokości nuty i jej dynamice. Na podstawie tych informacji generowana jest lista nut ze strukturą analogczną do tej, jaka użyta jest przy generowaniu plików MIDI opisanym w 5.7.

Każda z wynikowych nut transkrypcji algorytm porównujący klasyfikuje do jednej z dwóch grup:

- prawdziwie dodatnie (TP, z ang. *True positive*) gdy w utworze referencyjnym istnieje nuta o takim samym początku i czasie trwania ( $\pm$  margines błędu),
- fałszywie dodatnie (FP, z ang. *False positive*) gdy w utworze referencyjnym nie istnieje nuta o takim samym początku i czasie trwania ( $\pm$  margines błędu).

Zliczane są również wystąpienia fałszywego ujemnego (FN, z ang. *False negative*) sprawdzając ilość nut w utworze referencyjnym, których nie ma w wyniku transkrypcji. Przykład tej klasyfikacji pokazany jest na rysunku 5.1.



Rysunek 5.1: Przykład rolki pianina dwóch plików MIDI. Utwór referencyjny (lewa rolka) przedstawia rozłożenie nut (zielone prostokąty) w oryginalnej kompozycji. Na wyniku transkrypcji (prawa rolka) oznaczone są wykryte nuty (zielone prostokąty) wraz z klasyfikacją statystyczną (jako czerwone prostokąty pod wykrytymi nutami są oznaczone FN, czerwone prostokąty nad wykrytymi nutami oznaczają FP natomiast jasnozielone prostokąty reprezentują nuty z utworu referencyjnego, które były wykryte z marginesem błędów).

Funkcja porównująca przyjmuje jako argumenty maksymalny dozwolony błąd *maxError* będący liczbą sekund dozwolonej rozbieżności w czasie początku lub końca pomiędzy porównywanyimi zdarzeniami oraz dwie listy nut MIDI: *transcriptionMidiNotes* będącą wynikiem ewaluowanego algorytmu i *groundTruthMidiNotes* będącą oryginalną kompozycją. Działanie algorytmu można podzielić na trzy etapy:

1. policzenie wszystkich FP poprzez sprawdzenie, czy dana nuta z tablicy *transcriptionMidiNotes* posiada odpowiednik w tablicy *groundTruthMidiNotes*. Dla każdej pozycji bez pary zwiększa się licznik FP,
2. policzenie wszystkich TP i FN poprzez sprawdzenie, czy dana nuta z tablicy *groundTruthMidiNotes* posiada odpowiednik w tablicy *transcriptionMidiNotes*. Gdy para zostanie znaleziona zwiększa się licznik TP, w przeciwnym wypadku zwiększa się licznik FN,
3. policzenie miar statystycznych.

Szczegółowy opis wyliczanych miar statystycznych znajduje się w rozdziale 6.

W celach porównawczych zostało również przeprowadzone sprawdzenie czasu wykonywania poszczególnych algorytmów transkrypcji. Zagadnienie to jest szczególnie ciekawe w kontekście porównania czasu wykonania algorytmów wykorzystujących GPU do implementacji używających jedynie CPU. Zaimplementowany dekorator *profile* znajdujący się w pliku *server/utils/custom\_profile.py* każdorazowo mierzy czas wywołania funkcji i zapisuje go. Wynik przedstawiany jest w postaci średniej ( $\frac{\text{suma czasów}}{\text{ilość wywołań}}$ ) z wyłączeniem  $n$  naj wolniejszych i najszybszych wywołań w celu wykluczenia wartości skrajnych.

Podczas porównywania wyników algorytmów ważnym jest, aby dla każdego z nich dobrać parametry tak, aby wynikowe F1 było jak największe. Podczas gdy pojedyńczy parametr, jak np. długość okna, ma swoje odwzorowanie w manipulacji dokładności w dziedzinie czasu i częstotliwości, tak oszacowanie wektora optymalnych parametrów, z czego każdy może wpływać na pozostałe, nie jest tak trywialne. W celu znalezienia najlepszych argumentów zaimplementowanych algorytmów został napisany mechanizm, który iteruje po zbiorze kombinacji proponowanych argumentów i wylicza dla nich miarę F1 na podstawie zbioru walidacyjnego.

W celu poprawnej interpretacji ewaluacji metody zostały podzielone na dwie grupy:

1. przeznaczone jedynie dla sygnału monofonicznego
2. przeznaczone zarówno dla sygnału monofonicznego jak i polifonicznego.

Mechanizm porównujący wyniki algorytmów, którego implementacja znajduje się w pliku `server/utils/evaluate.py`, wykonuje operacje opisane w tej sekcji dla wszystkich algorytmów według schematu poniżej:

1. dla każdego algorytmu z badanej grupy wyznaczany jest wektor najlepszych argumentów na podstawie zbioru walidacyjnego (funkcja `validate_arguments` i wszystkich kombinacji argumentów ze zbioru zadeklarowanych przyjmowanych wartości (są to słowniki, których kluczami są nazwy przyjmowanych przez funkcje parametrów, a wartościami tablice ewaluowanych wartości),
2. dla każdego algorytmu badanej grupy wykonywana jest ewaluacja miar statystycznych i czasu wykonywania z wykorzystaniem zbioru testowego (funkcja `run_test_on_dataset_with_args`).

Dane wejściowe, na których wykonywana jest ewaluacja, deklarowane są jako zbiory danych opisane plikiem JSON. Każdy zbiór danych jest osobnym katalogiem w folderze `server/test_sounds/data_sets` zawierającym:

- plik o nazwie `metadata.json`, w którym opisany jest zbiór danych. Każdy element musi zawierać cztery parametry:
  - `split` - deklaracja do jakiej grupy podziału należy dana pozycja. Wykorzystywane wartości to `validation` (używane do wyznaczania najlepszych parametrów) i `test` (używane do ewaluacji),
  - `midi_filename` - relatywna względem katalogu zbioru danych ścieżka do pliku MIDI,
  - `audio_filename` - relatywna względem katalogu zbioru danych ścieżka do pliku audio,
  - `canonical_title` - nazwa kompozycji
- pliki audio i MIDI zgodne z opisem w `metadata.json`

Metody ewaluacyjne przyjmują nazwę zbioru danych, z którego mają być brane dane oraz generują kombinacje argumentów wejściowych na podstawie zadeklarowanych słowników `argsAc`, `argsAclos` itd. Ze względu na to, że złożoność obliczeniowa rośnie wy-

kładniczo wraz ze wzrostem możliwych wartości argumentów, algorytmy te wykorzystują multi-procesowość w celu polepszenia czasu wykonywania (co widać na listingu 5.13).

```
with concurrent.futures.ProcessPoolExecutor() as executor:
    processes = []
    bestF1 = 0
    bestArgs = None
    idx = 1
    for arg in possibleArgsCombinations:
        processes.append(executor.submit(validate_arguments, func, arg, evalObjects,
                                         saveRes, isResMidi, False, str(idx) + "/" + str(len(possibleArgsCombinations))))
        idx += 1
    for res in processes:
        currArg, currF1 = res.result()
        if currF1 > bestF1:
            bestF1 = currF1
            bestArgs = currArg

return currArg, bestF1
```

*Listing 5.13: Algorytm szukający najlepszej kombinacji argumentów dla zadanej funkcji func.*

## 5.9. Interfejs GUI

W ramach projektu został zaimplementowany interfejs graficzny, na który składa się serwer HTTP napisany w języku Python oraz strona napisana w języku *TypeScript* i technologii *React*. Daje ona możliwość wywołania każdej z wyżej opisanych implementacji algorytmów transkrypcji muzyki z wcześniej wyliczonymi optymalnymi parametrami.

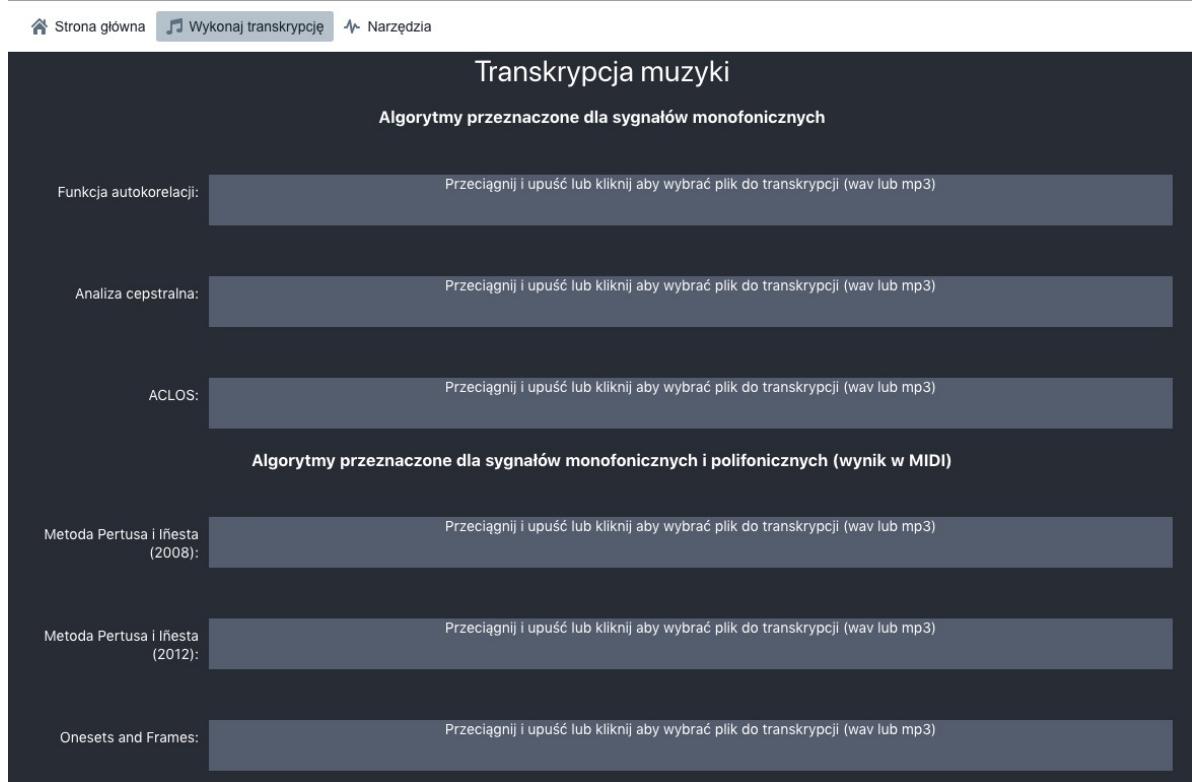
Działanie serwera sprowadza się do implementacji przechwytywania zapytań od klienta i odsyłania wyniku pożdanego algorytmu w jednym z dwóch postaci:

1. dla algorytmów przeznaczonych do analizy sygnałów monofonicznych - obiekt JSON, którego kluczami są nazwy poszczególnych wykresów (np. spektrogram, cepstrogram) a wartościami odpowiednie grafiki zakodowane do formatu Base64,
2. dla algorytmów przeznaczonych do analizy sygnałów monofonicznych i polifonicznych - plik MIDI wyniku transkrypcji.

Implementacja serwera znajduje się w pliku *server/server.py*.

Skrypty źródłowe graficznego interfejsu znajdują się w katalogu *static/src* i służą do wygenerowania statycznej strony, która jest wysyłana przez serwer do przeglądarki

klienta. Zbudowana paczka jest poddana procesowi minifikacji i składa się ze standar-dowych plików rozpoznawanych przez przeglądarki internetowe - *css*, *js* oraz *html*. Opis uruchomienia pełnego GUI na lokalnej maszynie znajduje się w pliku *README.md* w katalogu głównym projektu.



Rysunek 5.2: Zrzut ekranu widoku graficznego interfejsu zaimplementowanego w ramach projektu.

## 6. Ewaluacja

W tym rozdziale przedstawione są wyniki ewaluacji zaimplementowanych w ramach pracy magisterskiej algorytmów. Proces ewaluacyjny polega na znalezieniu wektora najlepszych argumentów danej funkcji, a następnie porównaniu wyniku transkrypcji z oryginalną kompozycją dla każdego utworu ze zbioru testowego. Ewaluacja zaimplementowanych algorytmów została wykonana przy pomocy metod przedstawionych w sekcji 5.8. Użyte do testów dane pochodzą z bazy MAESTRO [18]. Wynik przedstawiony jest w postaci miary statystycznej F1, dokładności oraz średniego czasu wykonywania.

Holistyczna metoda transkrypcji, opisana w rozdziale 4.3.1, nie jest ewaluowana, ponieważ wynik tego algorytmu nie jest porównywalny z pozostałymi tu opisanymi metodami ze względu na charakterystykę transkrypcji A2S, która zakłada inny poziom abstrakcji rezultatu.

Na podstawie mocy sklasyfikowanych przez algorytmy ewaluacyjne zbiorów TP, FP i FN mierzone są cztery miary statystyczne:

- dokładność (z ang. *accuracy*) reprezentująca skale poprawnie wykrytych obserwacji na tle wszystkich sklasyfikowanych obserwacji (zgodnie ze wzorem 6.1),
- precyzja (z ang. *precision*) reprezentująca jaka część klasyfikacji dodatniej jest prawdziwie dodatnia (zgodnie ze wzorem 6.2),
- czułość (z ang. *recall*) reprezentująca jaka część spośród wszystkich dodatnich wyników została wykryta przez transkrypcję (zgodnie ze wzorem 6.3),
- miara jakości F1 jako harmoniczna średnia precyzji i czułości (zgodnie ze wzorem 6.4).

$$\text{accuracy} = \frac{TP}{TP + FP + FN} \quad (6.1)$$

$$\text{precision} = \frac{TP}{TP + FP} \quad (6.2)$$

$$\text{recall} = \frac{TP}{TP + FN} \quad (6.3)$$

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (6.4)$$

Do ewaluacji algorytmów analizujących sygnał monofoniczny została wydzielona część danych bazy MAESTRO, a następnie zmodyfikowane zostały pliki MIDI tak, aby w danym momencie w czasie obecna była tylko jedna nuta. Na podstawie tych plików zostały wygenerowane sygnały audio z wykorzystaniem sampli pianina Steinway. Dane podzielone są do grupy używanej w celu wyznaczenia najlepszych argumentów oraz do grupy, na której wykonywane są testy. Ewaluacja algorytmów analizujących sygnał polifoniczny została wykonana na części bazy MAESTRO wybranej w sposób losowy z uwzględnieniem podziału danych (grupy testowe i ewaluacyjne). Pozycje oryginalnie przypisane do zbioru ewaluacyjnego wykorzystane zostały do wyznaczenia najlepszych argumentów, natomiast te ze zbioru testowego były użyte do wyliczenia wyników. Testy w obu przypadkach (dla sygnałów monofonicznych i polifonicznych) zostały wykonane na maszynie z procesorem Intel Core i5 3,19 GHz oraz kartą graficzną NVIDIA GeForce GTX 1060 3GB.

W tablicy 6.1 przedstawione są argumenty zaimplementowanych funkcji użyte podczas procesu ewaluacji dla sygnałów monofonicznych (analogicznie tablica 6.2 dla sygnałów polifonicznych). Nazwy skrócone algorytmów symbolizują:

- AC - autokorelacja 3.1
- ACLOS - autokorelacja na logarytmie widma 3.4
- CEPS - analiza cepstralna 3.3
- P&I 08 - metoda Pertusa i Iñesta z 2008 4.2.4
- P&I 12 - metoda Pertusa i Iñesta z 2012 4.2.5

Nazwy argumentów symbolizują:

- NO - długość okna w samplach,
- odstęp - odstęp pomiędzy kolejnymi oknami,
- marg. zer - wymiar tablicy zer dodanej do okna przed wykonaniem FFT,
- W - ilość analizowanych sąsiednich nut przy scalaniu podczas przetwarzania końcowego,
- min/max F0 - odpowiednio minimalna / maksymalna częstotliwość, jaka jest analizowana w kontekście F0 (w Hz),
- P - minimalny odstęp pomiędzy pikami w amplitudzie (w samplach),
- $\mu$  - próg amplitudy, poniżej którego zerowane są wartości,
- $\eta$  - minimalna ilość harmonicznych dla kandydata F0,

- $\gamma$  - maksymalna różnica w parametrze głośności w kombinacji,
- $f_r$  - maksymalna nieharmoniczność,
- C - maksymalna ilość równoległych dźwięków,
- lifter - mnożnik lifteringu. Dla wartości 0 funkcja lifteringu nie jest wykonywana,
- $k$  - ważność współczynnika gładkości,
- K - zakres wygładzania czasowego (w oknach czasowych),
- M - zakres śledzenia wysokości (w oknach czasowych).

Wartość „-” oznacza, że parametr nie jest wykorzystywany w danym algorytmie. Metody Pertusa i Iñesta posiadają więcej argumentów niż pozostałe algorytmy, więc przeznaczona została dla nich dodatkowa tabela, w której wymienione są parametry istotne tylko dla tych funkcji.

algorytm	NO	odstęp	marg. zer	W	min/max F0
AC	1024	1024	-	1	50/5500
ACLOS	4096	1024	2048	3	-
CEPS	4096	1024	8192	3	-
P&I 08	2048	512	8192	3	50/5500
P&I 12	2048	512	12288	3	50/5500

algorytm	$P$	$\mu$	$\eta$	$\gamma$	$f_r$	C	lifter	$k$	K	M
P&I 08	8	2	1	0.1	0.32	1	0	-	-	-
P&I 12	8	4	1	0.1	0.22	1	8	3	2	4

Tablica 6.1: Wektory najlepszych argumentów dla każdej funkcji transkrypcji dla sygnałów monofonicznych.

Argumenty użyte podczas ewaluacji sygnałów polifonicznych są przedstawione w tablicy 6.2 z analogiczną do opisanej wyżej dla sygnałów monofonicznych konwencją nazewnictwa.

algorytm	NO	odstęp	marg. zer	$W$	min/max F0					
P&I 08	2048	512	8192	3	50/5500					
P&I 12	2048	512	12288	3	50/5500					
	$P$	$\mu$	$\eta$	$\gamma$	$f_r$	C	lifter	$k$	K	M
P&I 08	8	2	1	0.1	0.32	1	0	-	-	-
P&I 12	8	4	1	0.1	0.22	1	8	3	2	4

Tablica 6.2: Wektory najlepszych argumentów dla funkcji transkrypcji dla sygnałów polifonicznych.

Wyniki ewaluacji algorytmów dla sygnałów monofonicznych znajdują się w tabeli 6.3. Testy wykonane zostały z maksymalnym błędem początku i trwania nuty równym 0.085 sekundy na 11 kompozycjach z łącznie 269 zdarzeniami MIDI i łącznym czasem odtwarzania 72s. Miary statystyczne są średnią miar wszystkich przypadków testowych. Kolumna T reprezentuje średni czas wykonania algorytmu w sekundach. Kolumny FN, FP i TP odpowiadają mocy zbiorów wszystkich sklasyfikowanych zdarzeń. W celu uzyskania bardziej rzetelnego rezultatu czasu wykonania wszystkie testy były wykonane dziesięciokrotnie i odrzuconych zostało 5 najszerszych i najwolniejszych wywołań.

algorytm	FN	FP	TP	dokładność	precyzja	czułość	F1	T
AC	106	229	163	0.326	0.380	0.515	0.425	3.96
ACLOS	104	357	165	0.266	0.297	0.485	0.360	11.82
CEPS	83	487	186	0.279	0.309	0.647	0.400	0.42
CEPS GPU	-	-	-	-	-	-	-	1.97
P&I 08	112	198	157	0.337	0.430	0.484	0.443	4.31
P&I 12	142	284	127	0.225	0.308	0.367	0.320	5.54
Onsets&Frames	122	158	147	0.299	0.360	0.401	0.379	9.45

Tablica 6.3: Wyniki transkrypcji sygnałów monofonicznych

Warto zwrócić uwagę na fakt, że metoda Pertusa i Iñesta z 2008 osiągnęła lepsze wyniki we wszystkich miarach jakościowych od znowelizowanej wersji. Możliwym wyjaśnieniem tego jest znacząca różnica w przetwarzaniu końcowym tych algorytmów, gdzie

wersja z 2012 roku używa wygładzania czasowego, które zaprojektowane było z myślą o transkrypcji kompozycji polifonicznych.

Podczas tak zdefiniowanej ewaluacji algorytm wykrywania F0 przy pomocy cepstrum liczonego na GPU jest wolniejsze od tego, liczonego na CPU. Dzieje się tak dlatego, że plan obliczeń GPU dla innych danych wejściowych musi być wyliczany w każdej iteracji. Dla przedstawienia realnego przyspieszenia, jakie daje użycie karty graficznej przy wyliczaniu cepstrum z sygnału zostały wykonane dodatkowe testy w postaci 1000 powtórzeń analizy F0 algorymem cepstrum z użyciem GPU i użyciem jedynie CPU dla pojedyńczego pliku dźwiękowego o czasie odtwarzania 38s. Różnica tych ewaluacji polega na tym, że plan obliczeń GPU jest wyliczany tylko dla pierwszego przypadku, a każde kolejne wywołanie używana wyliczonego wcześniej planu. W tym teście, z odrzuceniem 10 skrajnych wyników, średni czas wykonywania algorytmu na samym CPU wyniósł 0.828s. podczas gdy algorytm wykonany z użyciem GPU osiągnął średni czas wykonywania równy 0.670s, co daje w przybliżeniu 19% przyspieszenia obliczeń.

Wyniki ewaluacji algorytmów dla sygnałów polifonicznych znajdują się w tabeli 6.4. Testy wykonane zostały z maksymalnym błędem początku i trwania nuty równym 0.085 sekundy na 2 kompozycjach z łącznie 4431 zdarzeniami MIDI i łącznym czasem odtwarzania 310s. Miary statystyczne są średnią miar wszystkich przypadków testowych, natomiast kolumna T reprezentuje średni czas wykonania algorytmu w sekundach. Kolumny FN, FP i TP odpowiadają mocy zbiorów wszystkich sklasyfikowanych zdarzeń.

algorytm	FN	FP	TP	dokładność	precyzja	czułość	F1	T
P&I 08	1452	16023	2979	0.157	0.142	0.663	0.233	5250.99
P&I 12	1748	9198	2683	0.181	0.206	0.602	0.305	2675.14
Onsets&Frames	4125	3695	306	0.029	0.058	0.052	0.055	40.50

Tablica 6.4: Wyniki transkrypcji sygnałów polifonicznych

Z pozyskanych wyników można zaobserwować, że metoda Pertusa i Iñesta z 2008 roku jest w stanie wykryć więcej obserwacji TP od wersji z 2012 kosztem gorszej dokładności, precyzji i czasu. Jest to związane z innym przetwarzaniem końcowym, które w metodzie z 2012 roku dokładniej sprawdza kontekst wykrytych dźwięków. Większa wartość parametru dozwolonej nieharmoniczności algorytmu z 2008 roku doprowadziła do

powstania większej ilości kombinacji do przetworzenia, co znacząco zwiększa czas wykonywania transkrypcji.

Analizując przedstawione wyniki ewaluacji należy zwrócić uwagę na fakt, że ilość danych, na których podstawie został wyliczone wynik jest znacznie mniej niż w większości z cytowanych prac. Zaproponowana metoda klasyfikacji wykrytych dźwięków również różni się od tych, zaprezentowanych w innych pracach. Wiele nut zostało oznakowanych jako FN ze względu na mały margines błędu używany przy klasyfikacji. Transkrypcje często posiadały wykryte nuty oryginalnej kompozycji, lecz były przesunięte w czasie na tyle mocno, że klasyfikowały się jako FP.

## Zakończenie

W tej pracy magisterskiej przedstawiony został problem automatycznej transkrypcji muzyki wraz z metodologiami jego rozwiązania. W rozdziale 1 zdefiniowane zostały pojęcia teorii muzyki oraz jej cechy, jak i ich istotność w kontekście opisywanego tematu. W rozdziale 2 przedstawione zostały ogólne założenia transkrypcji muzyki wraz ze szczegółowym opisem transformaty Fouriera. W rozdziałach 3 i 4 opisane zostały wybrane algorytmy do wykrywania wysokości F0 w sygnałach dźwięków odpowiednio monofonicznych i polifonicznych. Kod, który powstał w ramach tej pracy implementuje przedstawione metody i opisany został w rozdziale 5. Zaimplementowane zostały również funkcje ewaluacyjne, które porównują omawiane algorytmy miarą F1 oraz mierzą ich czas wykonania. Opis ewaluacji zaimplementowanych metod transkrypcji muzyki opisany został w rozdziale 6.

Znacząca część pracy skupia się jedynie na problemie częściowym transkrypcji muzyki jakim jest wykrycie wysokości obecnych w sygnale dźwięków. Pozostałe problemy, jak analiza rytmiki czy barwy dźwięku zostały jedynie przedstawione, bez dogłębnej analizy. Przedstawione metodologie pokazują, że zadanie transkrypcji muzyki jest złożonym tematem przetwarzania sygnałów, który nie ma jednego optymalnego rozwiązania. Opisane algorytmy zakładają muzykę komponowaną w oparciu o system równomiernie temperowany, co jest istotnym ograniczeniem możliwych danych wejściowych.

Polifoniczność w sygnale jest czynnikiem, który najmocniej wpływa na złożoność problemu transkrypcji muzyki. Rozróżnienie poszczególnych równoległych dźwięków wykonywane jest przy pomocy różnych metod, które często łączone są w złożone algorytmy, jak w metodach Pertusa i Iñesta 4.2.4 4.2.5. Inną techniką transkrypcji sygnałów dźwięków polifonicznych jest wykrywanie wzorców w sygnale, co ma miejsce w algorytmach bazujących na uczeniu maszynowym. Przykładami takich rozwiązań są *Onsets and Frames* i holistyczna metoda Pertusa, opisane w rozdziałach odpowiednio 4.3.2 i 4.3.1.

W dniu pisania pracy nie istnieje rozwiązanie omawianego problemu dla uniwersalnych danych wejściowych. Co więcej, nie istnieje bezbłędny system transkrypcji nawet dla kompozycji z ograniczeniami do użycia pojedyńczego instrumentu i systemu równomiernie temperowanego. Analiza wyników algorytmów pokazuje jednak, że wraz z nowymi rozwiązaniami na przestrzeni lat osiągane są nowe rekordy pod względem jakości wyników algorytmów transkrypcji muzyki.

## **Spis rysunków**

1.1	Przykład notacji muzycznej wygenerowanej na podstawie pliku MIDI w programie Logic Pro X. Widać na nim takie informacje jak metrum czy wysokości i długości granych dźwięków . . . . .	6
1.2	Sygnal akustyczny z wyraźnymi metrykami muzycznymi. Na osi poziomej liczbowo oznaczona jest metryka nutowa (4 beaty), mniejszym zaś regularne odstępy taktowe. . . . .	8
1.3	Ilustracja przedstawiająca trzy oktawy klawiatury pianina z oznaczonymi muzycznymi nazwami nut . . . . .	8
1.4	Wykres przedstawiający sygnał dźwięku o zmiennej amplitudzie (głośności) względem czasu wygenerowany przez klasyczne pianino elektryczne. . . . .	12
1.5	Spektrogramy dźwięku C3 (130.81Hz) grane na czterech różnych instrumentach (flet, pianino, altówka i saksofon) pokazują różnice w ilości i rozmieszczeniu harmonicznych każdego z dźwięków. Spektrogramy wykonane z 4096 samplami na okno, odstępem okien 1024 sampli i funkcją okna Hanna. Jaśniejsze obszary symbolizują większą magnitudę.	13
2.1	Grafy przedstawiają 3 różne reprezentacje danych na średnim poziomie tego samego dźwięku trąbki grającej dźwięk C4. Lewy wykres przedstawia spektrogram na logarytmicznej skali częstotliwości. Środkowy wykres przedstawia reprezentację ścieżek sinusoidalnych, przy czym szerokość lini oznacza amplitudę poszczególnych sinusoid. Prawy wykres przedstawia prosty model słuchowy. Wykres z [28, s. 14], tłumaczenie własne. . . . .	17
2.2	Przykłady funkcji okna czasowego dla długości 2048 próbek. . . . .	22
2.3	Przykład analizy częstotliwości z wykorzystaniem STFT. Każdy z wykresów przedstawia to samo okno po uzupełnieniu na końcu zerami do długości $N_{FFT} = 128, 256, 1024$ (kolejno od góry). Wykres z [54, s. 224]. . . . .	24

3.1	Wynik autokorelacji na kompozycji „Oda do radości” IX symfonii Beethovena zagranej w sposób monofoniczny na pianinie. Użyto okno Hanna o długości 2048 sampli z odstępami długości 2048 sampli. Wybór F0 polegał na wybraniu największego współczynnika korelacji w oknie. Jaśniejsze obszary na korelogramie oznaczają większą magnitudę. . . . .	28
3.2	Spektrogramy sygnału sekwencji C3-D3-C3-B2 zagranej na pianinie wyliczone przy pomocy STFT z oknem Hanna długości odpowiednio 1024 i 4096 dla wykresów (a) i (b) z odstępami o długości 512 sampli. Jaśniejsze obszary oznaczają większą magnitudę. . . . .	29
3.3	Przykład pojedyńczego komponentu spektralnego ze spektrogramu z rysunku 3.2 o długości okna 4096. . . . .	30
3.4	Wynik cepstrum na na „Oddie do radości” IX symfonii Beethovena zagranej w sposób monofoniczny na pianinie. Użyto okno Hanna o długości 2048 sampli z odstępami długości 512 sampli. Wybór F0 polegał na wybraniu największego współczynnika cepstralnego w oknie. Jaśniejsze obszary w spektrogramach i cepstrogramie oznaczają większą magnitudę. . . . .	33
3.5	Schemat działania algorytmu ACLOS do wykrywania wysokości dźwięku. Schemat z [29, s. 233], tłumaczenie własne. . . . .	34
3.6	Wykres przedstawiający spektrum na pojedyńczym oknie przed i po nałożeniu lifterowania ze współczynnikiem równym 10 . . . . .	35
3.7	Wynik ACLOS użytego na kompozycji „Oda do radości” IX symfonii Beethovena zagranej w sposób monofoniczny na pianinie. Użyto okno Hanna o długości 2048 sampli z odstępami długości 512 sampli. Wykresy (c) i (d) przedstawiają wynik równania 3.7 na tym samym oknie po zastosowaniu lifteringu. Jaśniejsze obszary w cepstrogramie oznaczają większą magnitudę. . . . .	36
4.1	Funkcja autokorelacji 3.1 zastosowana na nagraniu Preludium op. 28 nr. 4 Fryderyka Chopina zagrany na pianinie w Em. Użyto okno Hanna o długości 2048 sampli z odstępami długości 2048 sampli. . . . .	38

4.2	Wynik funkcji autokorelacji zastosowanej na nagraniu akordu Em zagrany na pianinie. Użyto okno Hanna o długości 2048 sampli z odstępami długości 2048 sampli. Wykresy (c) jest reprezentacją wyników AC na 11 oknie (próbce danych od 20480 do 22528 sampli sygnału wejściowego). . . . .	39
4.3	Spektrogram nagrania Preludium op. 28 nr. 4 Fryderyka Chopina w Em zagraneego na pianinie. Użyto okna Hanna o długości 2048 sampli, marginesu zer o długości 2048 i odstępów długości 512 sampli. Jaśniejsze obszary w spektrogramie oznaczają większą magnitudę. . . . .	40
4.4	Spektralne komponenty nagrania akordu Em zagraneego na pianinie oraz trzech poszczególnych nut zagranych na tym samym pianinie nagranych osobno. Użyto okna Hanna o długości 2048 sampli i marginesu zer długości 2048. . . . .	41
4.5	Spektrogram nagrania akordu Em zagraneego na pianinie. Użyto okna Hanna o długości 2048 sampli z marginesem zer długości 2048 i marginesu zer długości 2048. Jaśniejsze obszary w spektrogramie oznaczają większą magnitudę. . . . .	42
4.6	Wynik analizy cepstralnej nagrania akordu Em zagraneego na pianinie. Użyto okno Hanna o długości 2048 sampli z marginesem zer długości 2048 i odstępami długości 2048 sampli. Jaśniejsze obszary w cepstrogramie oznaczają większą magnitudę. . . . .	42
4.7	Wynik analizy algorytmem ACLOS nagrania akordu Em zagraneego na pianinie. Użyto okno Hanna o długości 2048 sampli z marginesem zer długości 2048 i odstępami długości 2048 sampli. Jaśniejsze obszary w spektrogramach oznaczają większą magnitudę. . . . .	42
4.8	Zobrazowanie zasady gładkości spektrum. Logarytmiczne spektrum mocy zawierające dwa dźwięki. Widmo zostało przefiltrowane przez lifting wysoko przepustowy. Wykres z [26, s. 3383]. . . . .	47
4.9	Schemat metody z podziałem na metodę podstawową opisaną w [40] i rozszerzoną opisaną w [39]. Schemat z [39, s. 3], tłumaczenie własne. . .	53
4.10	Architektura sieci neuronowej zastosowanej w pracy [46]. Diagram z [46, s. 5] , tłumaczenie własne. . . . .	58

4.11	Diagram przedstawiający architekturę modelu Onesets and Frames.	
	Diagram z [16], tłumaczenie własne. . . . .	60
5.1	Przykład rolki pianina dwóch plików MIDI. Utwór referencyjny (lewa rolka) przedstawia rozłożenie nut (zielone prostokąty) w oryginalnej kompozycji. Na wyniku transkrypcji (prawa rolka) oznaczone są wykryte nuty (zielone prostokąty) wraz z klasyfikacją statystyczną (jako czerwone prostokąty pod wykrytymi nutami są oznaczone FN, czerwone prostokąty nad wykrytymi nutami oznaczają FP natomiast jasnozielone prostokąty reprezentują nuty z utworu referencyjnego, które były wykryte z marginesem błędu). . . . .	73
5.2	Zrzut ekranu widoku graficznego interfejsu zaimplementowanego w ramach projektu. . . . .	77

## **Spis tablic**

1.1	Wysokości kolejnych dźwięków A zgodna ze wzorem 1.1 . . . . .	10
2.1	Definicje funkcji okien czasowych $w(t)$ i analityczne wzory ich widm $W(\omega)$	22
6.1	Wektory najlepszych argumentów dla każdej funkcji transkrypcji dla sygnałów monofonicznych. . . . .	80
6.2	Wektory najlepszych argumentów dla funkcji transkrypcji dla sygnałów polifonicznych. . . . .	81
6.3	Wyniki transkrypcji sygnałów monofonicznych . . . . .	81
6.4	Wyniki transkrypcji sygnałów polifonicznych . . . . .	82

## Spis listingów

5.1 Przykład pętli po danych wejściowych z uwzględnieniem okna czasowego. Takie dzielenie danych, z nakładaniem funkcji okna i dodaniem marginesu zer jest wykonywane w większości zaimplementowanych algorytmów. . . . .	62
5.2 Wyliczanie maksymalnego i minimalnego przesunięcia funkcji autokorelacji . . . . .	63
5.3 Funkcja autokorelacji . . . . .	63
5.4 Wyliczanie cepstrum mocy . . . . .	64
5.5 Estymacja F0 przy pomocy cepstrum . . . . .	64
5.6 Proces wyliczania cepstrum przy użyciu GPU . . . . .	66
5.7 Filtrowanie w domenie quefrencji . . . . .	67
5.8 Wyznaczanie F0 w ACLOS . . . . .	67
5.9 Trzon algorytmu Pertusa i Iñesta (2008) . . . . .	69
5.10 Wyliczanie istotności w algorytmie Pertusa i Iñesta (2012) . . . . .	70
5.11 Funkcja wyliczająca wagę krawędzi w algorytmie śledzenia wysokości nut . . . . .	70
5.12 Funkcja wyliczająca wagę krawędzi w algorytmie śledzenia wysokości nut . . . . .	72
5.13 Algorytm szukający najlepszej kombinacji argumentów dla zadanej funkcji func. .	76

## Bibliografia

- [1] E. Benetos, S. Dixon, Z. Duan i S. Ewert. „Automatic Music Transcription: An Overview”. W: *IEEE Signal Process. Mag.* 36.1 (2019), s. 20–30. DOI: 10.1109/MSP.2018.2869928.
- [2] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff i A. Klapuri. „Automatic music transcription: challenges and future directions.” W: *Journal of Intelligent Information Systems* 41.3 (2013), s. 407–434. DOI: 10.1007/s10844-013-0258-3.
- [3] J. A. Bilmes. „Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm”. Prac. mag. Massachusetts Institute of Technology, 1993, s. 77–84.
- [4] B. P. Bogert, M.J.R. Healy i J.W. Tukey. „The quefrency analysis of time series for echoes : cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking”. W: *Time Series Analysis* (1963), s. 209–243.
- [5] E. M. Burns. „Inversal, scales and tuning”. W: *Diana Deutsch (ed.), The Psychology of Music* (1999), s. 215–264.
- [6] D. G. Childers, D. P. Skinner i R. C. Kemerait. „The Cepstrum: A Guide to Processing”. W: *Proceedings of the IEEE* 65.10 (1977), s. 1428–1443. DOI: 10.1109/proc.1977.10747.
- [7] E. F. Clarke. „Rhythm and timing in music.” W: *Diana Deutsch (ed.), Psychology of Music* (1999), s. 473–500.
- [8] J. Cooley i J. Tukey. „An Algorithm for the Machine Calculation of Complex Fourier Series”. W: *Mathematics of Computation* 19.90 (1965), s. 297–301. DOI: 10.1090/S0025-5718-1965-0178586-1.
- [9] M. Davy i S. J. Godsill. „Bayesian Harmonic Models for Musical Signal Analysis”. W: *Bayesian Statistics* 7 (2003), s. 1–10. URL: [https://www.researchgate.net/publication/2845155\\_Bayesian\\_Harmonic\\_Models\\_for\\_Musical\\_Signal\\_Analysis](https://www.researchgate.net/publication/2845155_Bayesian_Harmonic_Models_for_Musical_Signal_Analysis) (term. wiz. 09.09.2019).
- [10] E. W. Dijkstra. „A Note on Two Problems in Connexion with Graphs”. W: *Numerische Mathematik* 1.1 (1959), s. 269–271. DOI: 10.1007/BF01386390.
- [11] D. Eck. *Welcome to Magenta!* 2016. URL: <https://magenta.tensorflow.org/blog/2016/06/01/welcome-to-magenta/> (term. wiz. 12.01.2020).

- [12] V. Emiya, N. Bertin, B. David i R. Badeau. *MAPS - A piano database for multipitch estimation and automatic transcription of music*. 2010. URL: <https://hal.inria.fr/inria-00544155>.
- [13] B. Gold. „Computer Program for Pitch Extraction”. W: *The Journal of the Acoustical Society of America* 34.7 (1962), s. 916–921. DOI: 10.1121/1.1918221.
- [14] M. Goto i Y. Muraoka. „A Beat Tracking System for Acoustic Signals of Music”. W: *ACM International Conference on Multimedia* (1994), s. 365–372.
- [15] W. M. Hartmann. „Pitch, periodicity, and auditory organization”. W: *The Journal of the Acoustical Society of America* 100.6 (1996), s. 3491–3502. DOI: 10.1121/1.417248.
- [16] C. Hawthorne i E. Elsen. *Onsets and Frames: Dual-Objective Piano Transcription*. blog. 2018. URL: <https://magenta.tensorflow.org/onsets-frames>.
- [17] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. H. Engel, S. Oore i D. Eck. „Onsets and Frames: Dual-Objective Piano Transcription”. W: *CoRR* abs/1710.11153 (2017). arXiv: 1710.11153. URL: <http://arxiv.org/abs/1710.11153>.
- [18] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, Cheng-Zhi A. Huang, S. Dieleman, E. Elsen, J. Engel i D. Eck. „Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. W: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=r11YRjC9F7> (term. wiz. 12.01.2020).
- [19] H. Hermansky, N. Morgan i H.-G. Hirsch. „Recognition of speech in additive and convolutional noise based on RASTA spectral processing”. W: t. 2. 1993, s. 83–86. DOI: 10.1109/ICASSP.1993.319236.
- [20] Cheng-Zhi A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman i D. Eck. „An Improved Relative Self-Attention Mechanism for Transformer with Application to Music Generation”. W: *CoRR* abs/1809.04281 (2018). arXiv: 1809.04281. URL: <http://arxiv.org/abs/1809.04281>.
- [21] H. Kameoka, T. Nishimoto i S. Sagayama. „Separation of harmonic structures based on tied Gaussian mixture model and information criterion for concurrent sounds”. W: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2004, Montreal, Quebec, Canada, May 17-21, 2004*. IEEE, 2004,

- s. 297–300. DOI: 10.1109/ICASSP.2004.1326822. URL: <https://doi.org/10.1109/ICASSP.2004.1326822>.
- [22] P. Kardas. *Homerecording: dla kązdego*. 1 wyd. Warszawa: Piotr Kardas Music Production, 2015. ISBN: 978-83-941802-1-8.
- [23] P. Kardas. *Homerecording: level up*. 1 wyd. Częstochowa: Piotr Kardas Music Production, 2016. ISBN: 978-83-941802-4-9.
- [24] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt i G. Widmer. „On the Potential of Simple Framewise Approaches to Piano Transcription”. W: *CoRR* abs/1612.05153 (2016). arXiv: 1612.05153. URL: <http://arxiv.org/abs/1612.05153>.
- [25] A. Klapuri. „Signal Processing Methods for the Automatic Transcription of Music”. Prac. dokt. Tampere: Tampere University of Technology, 2004. ISBN: 952-15-1147-8.
- [26] A. P. Klapuri. „Multipitch estimation and sound separation by the spectral smoothness principle”. W: 5 (2001), s. 3381–3384. DOI: 10.1109/ICASSP.2001.940384.
- [27] A. P. Klapuri. „Multiple Fundamental Frequency Estimation Based on Harmonicity and Spectral Smoothness”. W: *IEEE Transactions on Speech and Audio Processing* 11.6 (2003), s. 804–816. DOI: 10.1109/TSA.2003.815516.
- [28] A. Klapuri i M. Davy. *Signal Processing Methods for Music Transcription*. 1 wyd. New York: Springer Science + Business Media LLC, 2006, s. 3–17, 21–27, 203–227, 238–250. ISBN: 978-0-387-30667-4.
- [29] N. Kunieda, T. Shimamura i J. Suzuki. „Robust method of measurement of fundamental frequency by ACLOS: autocorrelation of log spectrum”. W: *IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. (1996), s. 232–235. DOI: 10.1109/ICASSP.1996.540333.
- [30] M. Lahat, R. Niederjohn i D. Krubsack. „A Spectral Autocorrelation Method for Measurement of the Fundamental Frequency of Noise-Corrupted Speech”. W: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35.6 (1987), s. 741–750. DOI: 10.1109/TASSP.1987.1165224.
- [31] F. Lerdahl i R. Jackendoff. *A Generative Theory of Tonal Music*. 1 wyd. Londyn: The MIT Press, 1996. ISBN: 978-0262621076.
- [32] R. J. McAulay i T. F. Quatieri. „Speech Analysis/Synthesis Based on a Sinusoidal Representation”. W: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34.4 (1986), s. 744–754. DOI: 10.1109/TASSP.1986.1164910.

- [33] M. E. McIntyre, R. T. Schumacher i J. Woodhouse. „On the oscillations of musical instruments”. W: *The Journal of the Acoustical Society of America* 74.5 (1983), s. 1325–1345. DOI: 10.1121/1.390157.
- [34] J. A. Moorer. „On the Transcription of Musical Sound by Computer”. W: *Computer Music Journal* 1.4 (1977), s. 32–38.
- [35] A. M. Noll. „Cepstrum pitch determination”. W: *The Journal of the Acoustical Society of America* 41.2 (1967), s. 293–309. DOI: 10.1121/1.1910339.
- [36] *Onsets and Frames Transcription Colab*. URL: [https://colab.research.google.com/notebooks/magenta/onsets\\_frames\\_transcription/onsets\\_frames\\_transcription.ipynb](https://colab.research.google.com/notebooks/magenta/onsets_frames_transcription/onsets_frames_transcription.ipynb) (term. wiz. 25.04.2020).
- [37] A. V. Oppenheim. „Superposition in a class of nonlinear systems”. Prac. dokt. Massachusetts Institute of Technology, 1964.
- [38] A. V. Oppenheim i R. W. Schafer. „From frequency to quefrency: a history of the cepstrum”. W: *IEEE Signal Processing Magazine* 21.5 (2004), s. 95–106. DOI: 10.1109/MSP.2004.1328092.
- [39] A. Pertusa i J. Iñesta. „Efficient methods for joint estimation of multiple fundamental frequencies in music signals”. W: *EURASIP Journal on Advances in Signal Processing* 2012.1 (2012). DOI: 10.1186/1687-6180-2012-27.
- [40] A. Pertusa i M. J. Iñesta. „Multiple fundamental frequency estimation using Gaussian smoothness”. W: 2008, s. 105–108. DOI: 10.1109/ICASSP.2008.4517557.
- [41] K. Pochwalski. *Dyskretna transformata Fouriera*. URL: <https://elektronikab2b.pl/technika/3096-dyskretna-transformata-fouriera> (term. wiz. 09.09.2019).
- [42] D. Quenneville. „Automatic Music Transcription”. Prac. lic. Middlebury College, 2018.
- [43] L. Rabiner. „On the Use of Autocorrelation Analysis for Pitch Detection”. W: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.1 (1977), s. 24–33. DOI: 10.1109/TASSP.1977.1162905.
- [44] R. B. Randall. „A history of cepstrum analysis and its application to mechanical problems”. W: *Mechanical Systems and Signal Processing* 97 (2017), s. 3–19. DOI: 10.1016/j.ymssp.2016.12.026.

- [45] *Repozytorium Onsets and Frames*. URL: [https://github.com/tensorflow/magenta/tree/master/magenta/models/onsets\\_frames\\_transcription](https://github.com/tensorflow/magenta/tree/master/magenta/models/onsets_frames_transcription) (term. wiz. 25.04.2020).
- [46] M. A. Román, A. Pertusa i J. Calvo-Zaragoza. *A holistic approach to polyphonic music transcription with neural networks*. 2019.
- [47] C. S. Sapp. *humdrum-data*. URL: <https://github.com/humdrum-tools/humdrum-data.git> (term. wiz. 12.01.2020).
- [48] W. A. Schloss. „On the Automatic Transcription of Percussive Music - From Acoustic Signal to High-Level Analysis”. Prac. dokt. Stanford, CA: Stanford University, 1985.
- [49] M. Karjalainen T. Tolonen. „A computationally efficient multipitch analysis model.” W: *IEEE Transactions on Speech and Audio Processing* 8.6 (2000), s. 708–716. DOI: [10.1109/89.876309](https://doi.org/10.1109/89.876309).
- [50] D. Talkin. „A robust algorithm for pitch tracking (RAPT)”. W: *Speech Coding and Synthesis* (1995), s. 495–518.
- [51] C. Yeh i A. Röbel. „A new score function for joint evaluation of multiple F0 hypothesis”. W: *International Conference on Digital Audio Effects* (2004), s. 234–239.
- [52] C. Yeh, A. Roebel i X. Rodet. „Multiple Fundamental Frequency Estimation and Polyphony Inference of Polyphonic Music Signals”. W: *IEEE Transactions on Audio, Speech, and Language Processing* 18.6 (2010), s. 1116–1126. DOI: [10.1109/TASL.2009.2030006](https://doi.org/10.1109/TASL.2009.2030006).
- [53] R. J. Zatorre, P. Belin i V. B. Penhune. „Structure and function of auditory cortex: music and speech”. W: *RENDS in Cognitive Sciences* 6.1 (2002), s. 37–46.
- [54] T. P. Zieliński. *Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań*. 1 wyd. Warszawa: Wydawnictwa Komunikacji i Łączności sp. z o.o., 2005. ISBN: 978-83-206-1640-8.