Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

Katedra Telekomunikacji

Praca dyplomowa
inżynierska

Aplikacja do analizy złożonego sygnału dźwiękowego instrumentów muzycznych

Application for analyzing the compound sound signal of musical instruments

Imię i nazwisko:	Antoni Jankowski
Kierunek studiów:	Teleinformatyka
Opiekun pracy:	dr inż. Jarosław Bułat

2021

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.

Streszczenie

Praca opisuje aplikację transponującą sygnał muzyczny do notacji MIDI, moduły składające się na nią, a także badania sygnału przeprowadzone przez autora i wybrane metody analizy sygnału. Celem systemu jest ułatwienie wejścia w świat muzyki elektronicznej muzykom związanym z tradycyjnymi instrumentami strunowymi takimi jak gitara, czy skrzypce.

Spis treści

Rozdział 1

Wprowadzenie

1.1 Pomysł i motywacja

Instrumenty muzyczne w najbardziej ogólnej klasyfikacji są podzielone na analogowe i cyfrowe. Na potrzeby tej pracy grupę instrumentów analogowych ograniczono do instrumentów strunowych, które produkują sygnał poprzez wprowadzanie w drganie struny lub okresowe zaburzenie pola elektromagnetycznego zazwyczaj wynikające z jego interakcji ze wspomnianą wcześniej struną. Przedstawicielami tej rodziny są m.in. gitary elektryczna, gitary akustyczna i skrzypce.

Do rodziny instrumentów cyfrowych zaliczane są instrumenty klawiszowe (tzw. keyboardy), pianina cyfrowe, trąbki i flety cyfrowe. Są to tak naprawdę interfejsy do wprowadzania informacji o tym jaki dźwięk jest grany w danej chwili, jaka jest jego moc i długość. Te informacje są zazwyczaj następnie interpretowane przez moduł generujący dźwięk o pożądanych cechach.

Współczesna muzyka opiera się głównie na instrumentach cyfrowych i brzmieniach generowanych za pomocą oprogramowania komputerowego. Instrumenty analogowe powoli odchodzą w zapomnienie, a razem z nimi muzycy z nimi związani.

Zadaniem systemu opisanego w tej pracy jest przeniesienie instrumentów analogowych do dziedziny cyfrowej, tak aby instrument analogowy mógł być interfejsem do modułu generującego dźwięk. System ma określić grane dźwięki a następnie przedstawić je w notacji czytelnej dla oprogramowania syntezującego brzmienia. Taki zabieg drastycznie zwiększy możliwości gitary i skrzypiec, pozwalając im nadążyć za dzisiejszym przemysłem muzycznym, który zdecydowanie przesunął się w stronę dźwięku cyfrowego.

1.2 Planowane rezultaty i wdrożenie

1.2.1 Plan

Zbudowanie aplikacji transponującej sygnał gitarowy do dziedziny cyfrowej można podzielić na dwa rodzaje w zależności od rodzaju rozpoznawanych dźwięków:

- **system monofoniczny** - rozpoznaje pojedyńcze dźwięki
- **system polifoniczny** - rozpoznaje wielodźwięki i akordy

W tym projekcie rozważane są zagadnienia dotyczące obydwu systemów, jednak tylko system monofoniczny został zaimplementowany.

Oczekiwany efekt końcowy to oprogramowanie czasu rzeczywistego i nierzeczywistego spełniające kryteria:

- pobieranie dźwięku z **zewnętrznej karty dźwiękowej** posiadającej wysokiej jakości przetworniki A/D,
- transpozycja **pojedynczych dźwięków** do dziedziny cyfrowej,
- latencja(opóźnienie) nie przeszkadzająca w graniu na instrumencie (dotyczy systemu czasu rzeczywistego),
- możliwość podłączenia aplikacji do modułu syntezującego dźwięk,
- kompatybilność aplikacji z systemami operacyjnymi **Linux** i Windows.

1.2.2 Wdrożenie

Proces budowy systemu rozpoczęty został przez badanie charakterystyki brzmienia gitary, więcej na ten temat w rozdziale 4. Po ustaleniu podstawowych właściwości sygnału autor podjął się próbnej implementacji pierwszego modułu - bramki szumów. Kolejnym etapem była implementacja pozostałych modułów i połączenie ich w aplikację czasu nierzeczywistego i rzeczywistego.

Aplikacja czasu nierzeczywistego została zbudowana z potrzeby analizy każdej kolejnej ramki sygnału i zachowania algorytmów. Docelowym projektem jest system czasu rzeczywistego mogący być wykorzystany na przykład na scenie lub w studio.

Rozdział 2

Automatyczna transkrypcja sygnału dźwiękowego

2.1 Koncepcja transkrypcji

Tradycyjnie w muzyce **transkrypcja** to *opracowanie utworu muzycznego na inny niż w oryginale zespół wykonawczy (orkiestracja) lub inny instrument*, a także *zapisanie utworu inną notacją muzyczną* [1]. Popularne jest przepisywanie utworów na przykład z pianina na gitarę. Taka operacja wymaga wiedzy o technikach gry stosowanych na obydwu instrumentach i wiedzy o tym, jakie kombinacje dźwięków są możliwe do zagrania - każdy instrument ma swój własny zestaw unikatowych cech. Inną popularną transkrypcją jest zmiana notacji z nut na tabulaturę gitarową.

W tej pracy transkrypcja polega na zapisaniu w innej notacji nie całego utworu, a jedynie fragmentu dźwięku. Obraną notacją jest standard MIDI opisany w następnej sekcji tego rozdziału.

2.2 Standard MIDI

MIDI, czyli *Musical Instrument Digital Interface* to standard opisujący protokół, interfejs cyfrowy i złącza służące do łączenia ze sobą szerokiej gamy instrumentów elektronicznych i komputera. Dzięki MIDI możliwa jest komunikacja między tymi instrumentami.

W tej sekcji zostanie wyjaśnione tylko minimum informacji o MIDI potrzebnych do zrozumienia dalszych konceptów.

Aby osiągnąć komunikację między komputerem a instrumentem elektronicznym obsługującym standard MIDI należy określić określić nr **kanału MIDI** wspólny dla obu stron, rodzaj i wartość wiadomości midi, a następnie ją przesłać.

2.2.1 Kanał MIDI

Kanał MIDI to wirtualny tunel przesyłający wiadomości MIDI. Każde urządzenie MIDI może otrzymywać i wysyłać wiadomości na nie więcej niż 16 kanałach. Jeżeli dwa instrumenty midi są ze sobą połączone fizycznie kablem MIDI, lub wirtualnie (instrumenty są uruchomione na jednym komputerze) i mają włączony ten sam kanał MIDI, to możliwa jest między nimi komunikacja.

2.2.2 Wiadomość MIDI

Wiadomość MIDI to połączenie informacji o wybranym kanale, statusie dźwięku (on/off), numeru granego tonu i jego głośności. MIDI definiuje także inne wartości możliwe do ustawienia poprzez wiadomości MIDI, jednak jest to wiedza wybiegająca poza potrzeby tej pracy.

- **Dźwięki w MIDI.** MIDI rozróżnia jedynie półtony i opisuje je liczbami naturalnymi 0-127. Jedna oktawa jest opisana 12 kolejnymi liczbami. Najniższy ton zdefiniowany przez standard o częstotliwości 8.18 Hz ma przypisany numer 0 w notacji MIDI, najwyższy ton to G9 o częstotliwości 12543.85 Hz i numerze 127 w notacji MIDI. W tej pracy założony został przedział tonów E2-E6 (40 - 88 w notacji MIDI), którego granice są odpowiednio najniższym i najwyższym dźwiękiem, które można zagrać na gitarze 24-progowej w strojeniu standardowym EADGBE.
- **Status** - to informacja o tym czy dźwięk jest grany, czy właśnie przestał być grany. Dany dźwięk jest grany od momentu otrzymania wiadomości ze statusem 'on' przypisanej do tego dźwięku, aż do otrzymania wiadomości ze statusem 'off'.

2.3 Przegląd istniejących rozwiązań

Obecnie są dostępne dwa produkty, które spełniają przedstawione wcześniej wymagania. MIDI Guitar 2 (MG2) od Jam Origin [2] i MiGiC [3]. MG2 to oprogramowanie rozpoznające pojedyncze dźwięki i akordy z bardzo dużą dokładnością. Można stwierdzić, że jest to najbardziej zaawansowany produkt w tej dziedzinie.
MiGiC, czyli MIDI Guitar Converter to mniej zaawansowana alternatywa dla MG2, jest to system jedynie monofoniczny.

Rozdział 3

Wybrane metody analizy sygnału muzycznego

W tym rozdziale opisane są znane autorowi pracy metody analizy sygnału, rozważane przy budowie systemu transkrypcji. Metody podzielono adekwatnie do modułów systemu, w których zostały użyte - metody detekcji uderzenia i określenia harmonicznej fundamentalnej. Drugim wyraźnym podziałem jest rozróżnienie w zależności od dziedziny sygnału na której dana metoda operuje:

- dziedzina czasu - sygnał muzyczny naturalnie występuje w dziedzinie czasu. Z sygnału można odczytać informacje o jego okresowości i energii. Udział konkretnych częstotliwości w sygnale nie jest oczywisty. Może się wydawać, że zaletą jest brak konieczności wykonania obliczeń w celu zmiany dziedziny sygnału, jednak na ogół aby otrzymać wartościowe informacje należy wykonać na sygnale funkcję autokorelacji.
- dziedzina częstotliwości - wymaga transformacji sygnału z dziedziny czasu za pomocą *DFT*. W przystępny sposób przedstawia informacje o częstotliwościach składających się na sygnał.

3.1 Analiza sygnału w dziedzinie częstotliwości

Widmo w przystępny sposób przedstawia udział danych częstotliwości w sygnale. Główny wpływ na jakość widma ma **rozdzielczość częstotliwościowa** - odstęp między kolejnymi dyskretnymi puntami widma. Rozdzielczość jest zależna od długości analizowanego sygnału N i jego częstotliwości próbkowania f_s w sposób opisany zależnością (3.1):

$$f_r = N/f_s \tag{3.1}$$

gdzie f_r to rozdzielczość częstotliwościowa widma. Zaletą widma częstotliwościowego jest f_r stała dla całego spektrum, w przypadku dziedziny czasu odpowiednik f_r jest

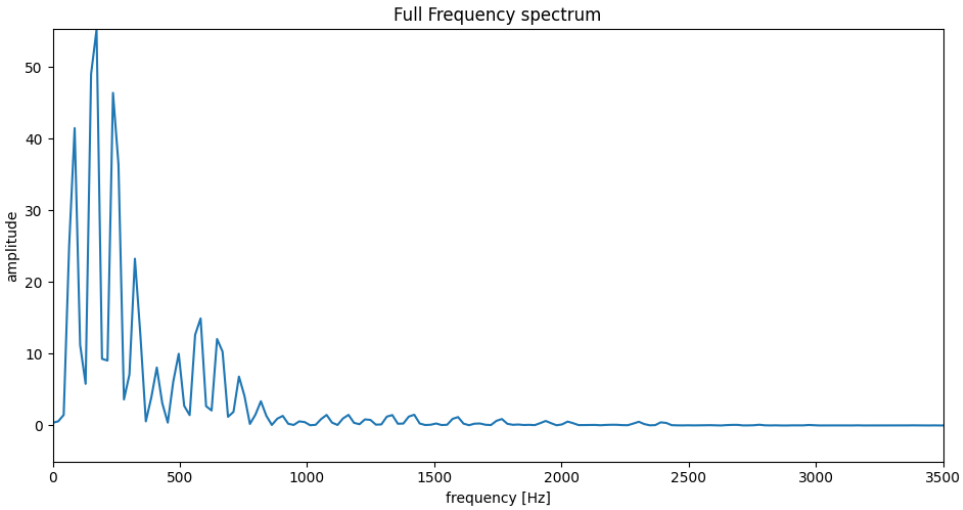
inny dla każdej częstotliwości. Z drugiej strony stała wartość f_r ogranicza dokładność niskich częstotliwości, o czym więcej w sekcji 3.1.2.

3.1.1 Analiza widma - wybór odpowiedniej skali

Popularne twierdzenie mówi, że człowiek słyszy odległości między tonami, jak i ich amplitudy w skali logarytmicznej [4]. Korzystając z tej obserwacji programy do edycji dźwięku przedstawiają widmo sygnału w skali logarytmicznej zarówno na osi częstotliwości jak i amplitudy, tak aby wykresy były możliwie najbardziej naturalne w odbiorze dla inżynierów dźwięku. W przypadku tej pracy, ustalenie skali logarytmicznej na osi częstotliwości byłoby niewygodne, znacznie łatwiej jest analizować widmo, na którym odstępy miedzy składnikami szeregu harmonicznego są równe. Wybranie odpowiedniej skali na osi amplitudy jest mniej oczywiste. Rozważmy dwa scenariusze, amplituda w skali liniowej (rys. 3.1) i amplituda w skali logarytmicznej (rys. 3.2).

Skala liniowa

Ten typ wykresu pozwala od razu wyróżnić najbardziej istotne dla sygnału składowe, jednak znalezienie długiego szeregu harmonicznych nie jest łatwe. Harmoniczne o niskich amplitudach mogą zostać pomylone z szumem.

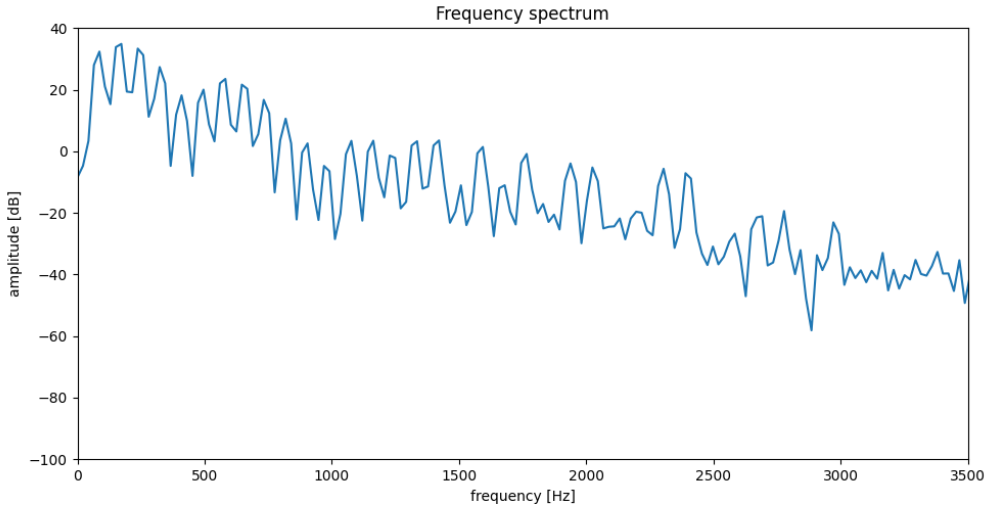


Rysunek 3.1: W skali liniowej na wykresie wyraźnie widać zaledwie kilka pierwszych harmonicznych.

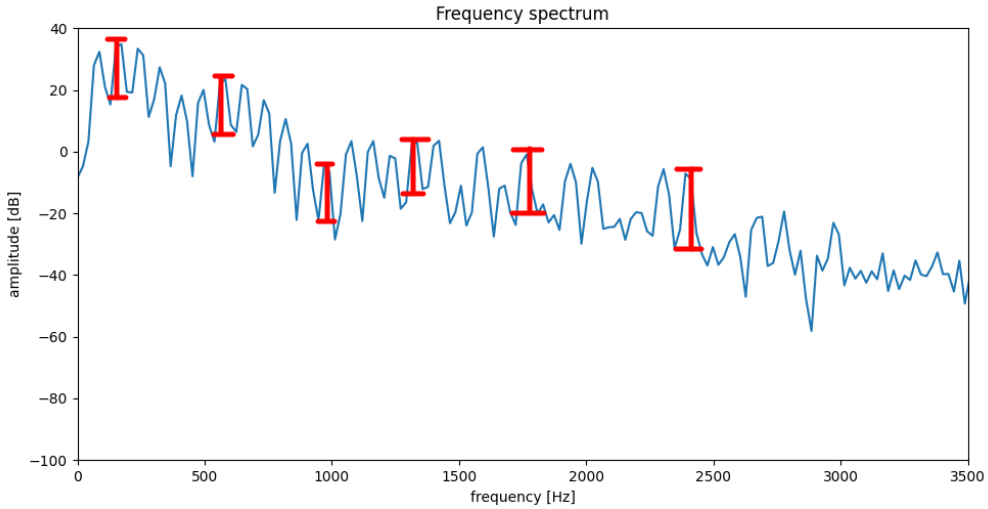
Skala logarytmiczna

Skala logarytmiczna zmniejsza różnice między harmonicznymi o wysokich i niskich amplitudach. Kolejną cechą wartą uwagi jest **amplituda względna** - odległość

między szczytem amplitudy harmonicznej, a jej podstawą (rys. 3.3). W skali logarytmicznej ta wartość określa ile razy szczyt jest większy od podstawy i jest podobna dla wszystkich składowych szeregu harmonicznego.



Rysunek 3.2: Zastosowanie skali logarytmicznej sprawia że różnice między silnymi i słabymi harmonicznymi są mniejsze.



Rysunek 3.3: Na rysunku zaznaczono przykładowe pomiary odległości szczytu od podstawy harmonicznej.

3.1.2 Szereg Harmoniczny

Jedną z cech charakterystycznych instrumentów strunowych takich jak gitara jest to, że na jeden dźwięk składa się wiele harmonicznych - częstotliwość fundamentalna i jej kolejne wielokrotności. Szereg widać na rys. 3.6. Znajomość szeregu rozpoczynającego się od f_0 pozwala określić jej wartość ze znacznie zwiększoną dokładnością. To dzięki temu, że wysokie częstotliwości lepiej wpasowują się w widmo częstotliwościowe niż niskie. Rozdzielczość widma jest stała dla wszystkich częstotliwości, ale wartość interwałów między tonami w Hercach rośnie wraz z wysokością dźwięku.

Odległość muzyczna a $\sqrt[12]{2}$

$\sqrt[12]{2}$ to stosunek częstotliwości między sąsiadującymi tonami, na przykład $C4 \approx 261.62$ i $C\#4 \approx 277.18$, odpowiada odległości muzycznej równej jednemu półtonowi.

$$\frac{C\#4}{C4} \approx \frac{277.18}{261.62} = 1.059 \approx \sqrt[12]{2} \tag{3.2}$$

Łatwo zauważyć, że ta odległość w przeliczeniu na Herce rośnie liniowo wraz z wysokością tonu i dla E2 (82.4 Hz) i F2 (87.3 Hz) wynosi tylko 4.9 Hz, a dla E5 (659.26 Hz) i F5 (698.46 Hz) jest to już aż 39.2 Hz, czyli prawie tyle co szerokość dwóch prążków widma o długości $N = 2048$ przy próbkowaniu $f_s = 44.1kHz$. W niskich częstotliwościach na jeden prążek przypada wiele tonów, natomiast w wysokich częstotliwościach nawet kilka prążków mieści się w granicach pasma jednego tonu. Tę zależność warto wykorzystać przy estymacji f_0 i zwiększyć wagę pomiarów wysokich harmonicznych.

Metoda estymacji f_0

Po interpolacji każdej z harmonicznych z osobna metodą **Bertecco-Yoshida BY1** [5] należy ponownie zwiększyć dokładność korzystając z szeregu harmonicznych. Należy ustalić taką wartość f_0 dla której suma (3.3) jest najmniejsza.

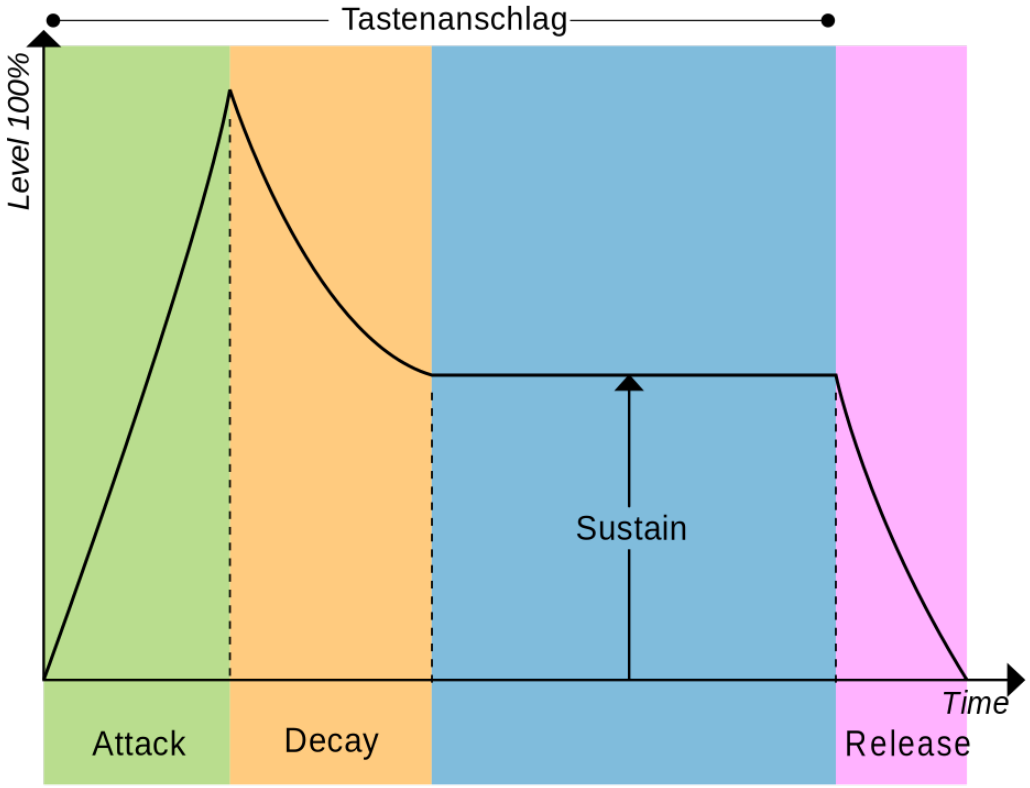
$$\sum_{n=0}^{M-1} \left| (\tilde{f_0} * (n+2) - Harm[n]) \right| \tag{3.3}$$

$\tilde{f_0}$ - badana estymowana wartość f_0
$Harm$ - lista częstotliwości kolejnych harmonicznych
M - długość Harm

3.2 Detekcja uderzenia

Znajomość chwili uderzenia w strunę pozwala ograniczyć analizowany sygnał o szumy, co wiąże się ze zwiększeniem dokładności dalszych pomiarów. Rozróżnienie następujących po sobie uderzeń jest ważne dla zachowania naturalności dynamiki dźwięku. W przebiegu uderzenia w strunę można wyróżnić 4 etapy (rys. 3.4):

- attack (atak) - moment uderzenia, sygnał osiąga maksymalną amplitudę,
- decay - amplituda opada od poziomu maksymalnego do poziomu podtrzymania,
- sustain (podtrzymanie) - amplituda przez dłuższą chwile pozostaje stała, a sygnał wybrzmiewa,
- release (zanikanie) - amplituda opada z poziomu podtrzymania do 0, zakończenie uderzenia.



Rysunek 3.4: Przedstawienie kolejnych etapów obwiedni według modelu **ADSR** [6]

Do określenia początku i końca uderzenia koniecznie jest rozpoznanie etapów **attack** i **release**, można do tego wykorzystać opisane poniżej metody badania zmienności funkcji energii sygnału (3.2.1) i wariancji widma (3.2.2).

3.2.1 Badanie zmienności funkcji energii sygnału

Energia sygnału dyskretnego, opisana wyrażeniem 3.4, rośnie i opada wraz ze zmianami amplitudy sygnału. Obserwując zmiany wartości energii można ustalić, w jakim etapie wybrzmiewania jest analizowany sygnał. Stały wzrost energii następujących

po sobie fragmentów sygnału oznacza **attack**, natomiast stały i zdecydowany spadek oznacza **release**.

$$E = \sum_{m=0}^{N} x[m]^2 \tag{3.4}$$

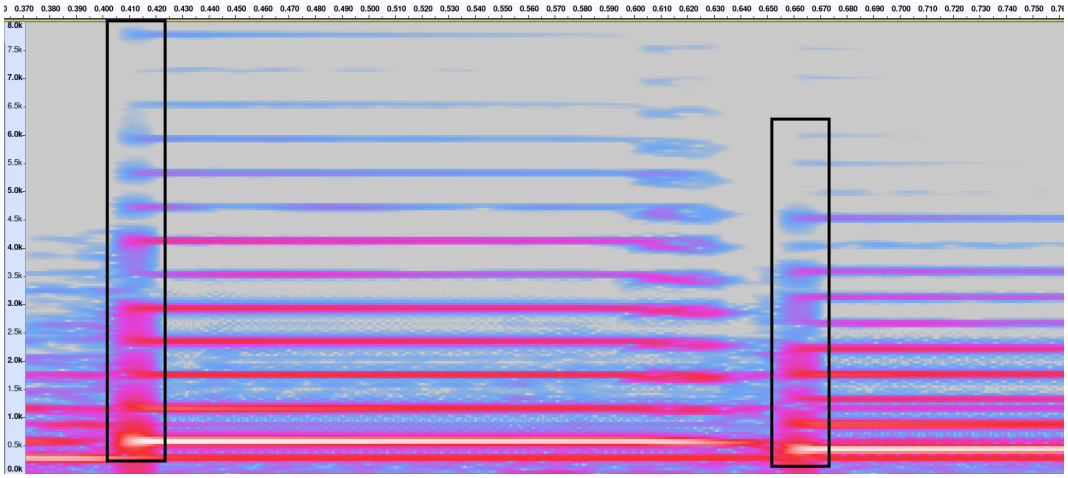
E - energia sygnału
x - analizowany sygnał
N - długość analizowanego sygnału

3.2.2 Detekcja uderzenia a wariancja widma

Uderzenie w struny można analizować również w dziedzinie częstotliwości. **Attack** charakteryzuje się wysoką energią, rozlaną po całym widmie. Następnym etapem wybrzmiewania jest **decay**. Energia sygnału stabilizuje się skupiając wokół harmonicznych, jak na rysunku 3.5. Metryką, która pozwoli rozróżnić **attack** od reszty stanów sygnału jest **wariancja**. Wariancja to klasyczna miara zmienności, jest średnią wartością sumy kwadratów odchyleń kolejnych wartości od średniej wszystkich wartości w danym zbiorze 3.5. Wartość wariancji rośnie wraz ze stabilizacją energii sygnału, to oznacza że dla stanu **attack** jej wartość będzie większa niż dla stanu **decay**.

$$s^2 = \sum_{i=0}^{N} \frac{(x_i - \overline{x})}{N-1} \tag{3.5}$$

s^2 - wariancja
x_i - wartość prążka widma o indeksie i
$\overline{x}$ - średnia wartość widma
N - długość widma



Rysunek 3.5: Uderzenie charakteryzuje się rozproszeniem energii po całym widmie. Obszar w czarnej ramce przedstawia początek uderzenia.

3.3 Wyznaczenie częstotliwości fundamentalnej

Częstotliwość fundamentalna f_0 to najniższa częstotliwość z sumy częstotliwości powiązanych harmonicznie. Kolejne powiązane z nią harmoniczne są jej wielokrotnością. W przypadku instrumentów strunowych f_0 definiuje jaki ton jest słyszalny dla ludzkiego ucha. Istnieją różne metody wyznaczania częstotliwości fundamentalnej, zaczynając od prostego wyznaczenia maksimum widma częstotliwościowego sygnału, aż po złożenie kilku metod w celu zwiększenia dokładności pomiaru. Rozwiązania brane pod uwagę przez autora pracy opisano poniżej.

3.3.1 Analiza funkcji autokorelacji

Funkcja autokorelacji sygnału dyskretnego jest dana wzorem (3.6). Opisuje podobieństwo sygnału do jego kopi dla danej wartości przesunięcia [7]. Podstawowym zadaniem autokorelacji jest wyznaczenie okresowości sygnału. Maksima autokorelacji sygnału okresowego znajdują się w punktach równych wielokrotności jego okresu. W przypadku sygnału instrumentu strunowego maksimum powinno występować w miejscu okresu oczekiwanej harmonicznej fundamentalnej. Wartość częstotliwości f_0 można określić dla sygnału x o częstotliwości próbkowania f_s korzystając z formuły (3.7).

$$R(x,l) = \sum_{n=0}^{N-1} x(n) * x(n+l) \tag{3.6}$$

x - analizowany sygnał
N - długość sygnału x
l - wartość przesunięcia x sygnału względem siebie

$$
\begin{aligned}
&xcorr = R(x,l), \quad dla \quad l = 0,1,2,...,N-1 \\
&maxInd = max(xcorr) \\
&f = Fs/maxInd
\end{aligned}
\tag{3.7}
$$

gdzie $f[Hz]$ to częstotliwość harmonicznej fundamentalnej.

Standardowa funkcja autokorelacji sygnału w dziedzinie czasu jest odporna na szum w sygnale, jednak nie zawsze jej maksimum znajduje się w punkcie równym okresowi f_0, szczególnie w przypadku gdy wybrzmiewają jeszcze częstotliwości z poprzedniego uderzenia. Autokorelacja jest bardzo kosztowana obliczeniowo ze względu na N^2 mnożeń dla sygnału o długości N. Można ten problem obejść stosując mnożenie w dziedzinie częstotliwości zamiast splatania sygnału w dziedzinie czasu.

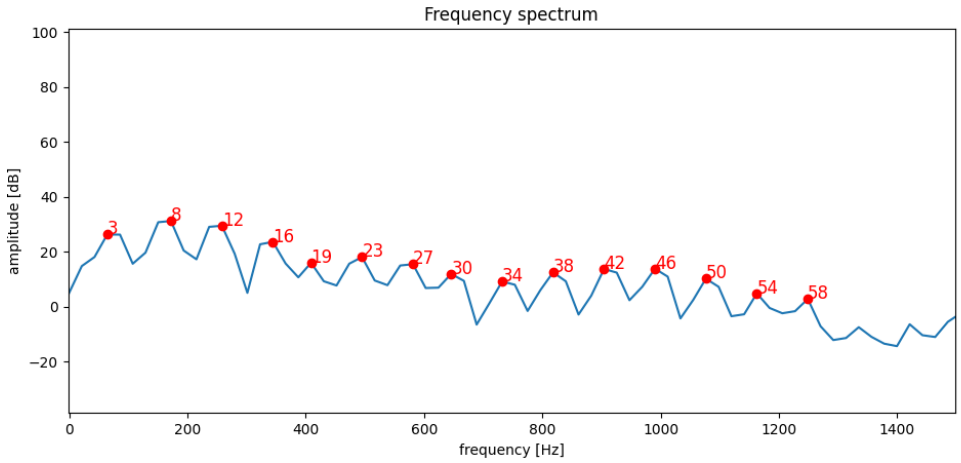
Dokładność pomiaru wartości f_0 rośnie wraz z częstotliwością próbkowania sygnału f_s. Wartość f_0 jest równa odwrotności znalezionego okresu $T = maxInd * \Delta t$, gdzie Δt to odległość w czasie między następującymi po sobie próbkami sygnału dyskretnego. Należy zauważyć że $\Delta t = 1/f_s$. Z tego powodu przy niskich częstotliwościach próbkowania wartość f_0 można określić jedynie orientacyjnie, szczególnie dla wysokich dźwięków.

3.3.2 EAC - Enhanced Autocorrelation

EAC [8], jak nazwa mówi rozwiązuje problemy standardowej autokorelacji. Autorzy zaproponowali metodę usunięcia maksimów lokalnych w punktach wielokrotności okresu f_0 opartą na różnicy wyniku standardowej autokorelacji z nadpróbkowaną kopią. Po tej operacji wszystkim wartością mniejszym od 0 przypisane jest 0. Cykl może być powtórzony kilka razy. W efekcie *EAC* daje jakościowo znacznie lepszy wynik niż standardowa funckja autokorelacji.

3.3.3 Maksimum widma częstotliwościowego

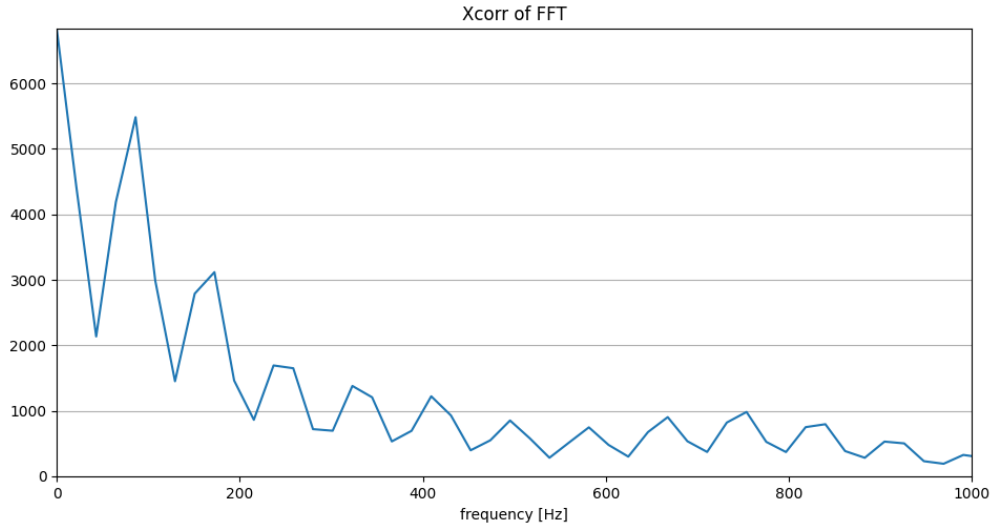
Podstawową techniką analizy sygnału w dziedzinie częstotliwości jest analiza widma częstotliwościowego w poszukiwaniu jego maksimów. Oczywiste wydaje się założenie, że prążek widma o najwyższej energii wskazuje na częstotliwość fundamentalną. W takim wypadku wystarczyłoby określić maksimum energii widma i odpowiadający mu prążek.
Faktycznie w wielu przypadkach to założenie jest prawdziwe, szczególnie dla wyższych dźwięków i już ustabilizowanego sygnału. Niestety popularne są również sytuacje w których to druga lub nawet trzecia harmoniczna mają większą energię od pierwszej - przykład poniżej (rys. 3.6). Poleganie na tej metodzie zazwyczaj kończy się na przeskakiwaniu znalezionej częstotliwości fundamentalnej miedzy pierwszymi dwoma harmonicznymi sygnału.



Rysunek 3.6: Prążki widma 8 i 12 mają wyraźnie większą energię niż prążek 3.

3.3.4 AFFT - Autokorelacja widma częstotliwościowego

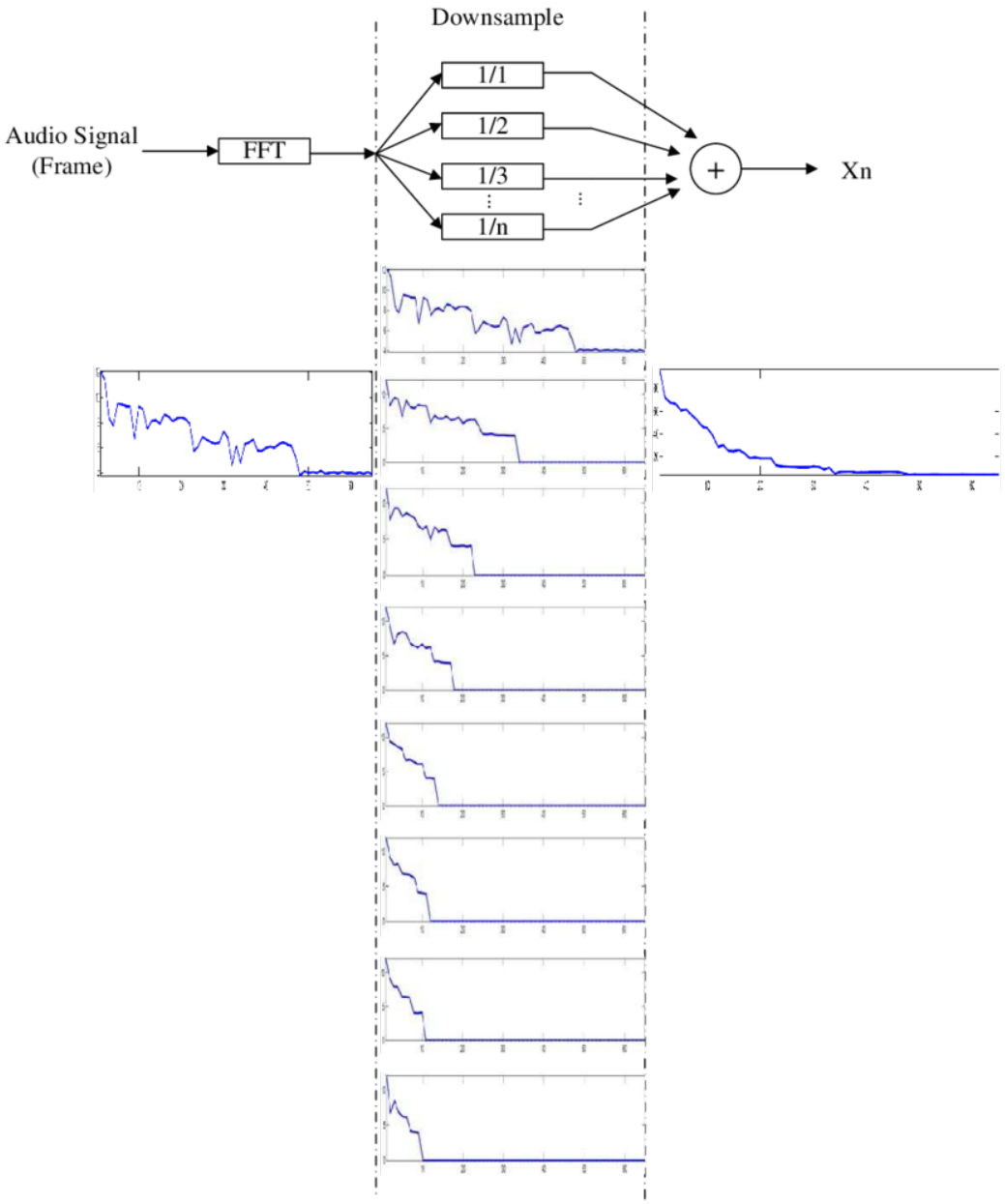
Funkcja autokorelacji zwraca informację o tym po jakim opóźnieniu sygnał jest najbardziej podobny do samego siebie. W dziedzinie częstotliwości funkcja autokorelacji określa najbardziej popularne odległości między kolejnymi prążkami widma, tym samym wskazując najbardziej istotne częstotliwości sygnału. Ta metoda w większości przypadków jest bezbłędna - tak jak na rysunku 3.7, jednak zdarza się, że to druga harmoniczna sygnału ma większą wartość funkcji autokorelacji. Wadą tego rozwiązania, tak jak w przypadku standardowej autokorelacji, jest wysoka złożoność obliczeniowa.



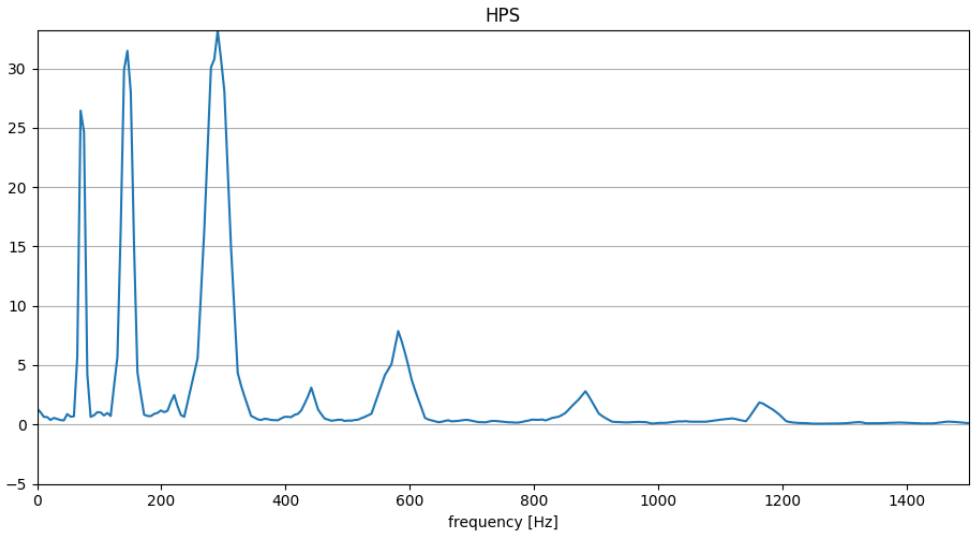
Rysunek 3.7: Zgodnie z oczekiwaniami, funkcja autokorelacji widma sygnału w okolicach 90 Hz osiąga największe maksimum lokalne.

3.3.5 HPS - Harmonic Product Spectrum

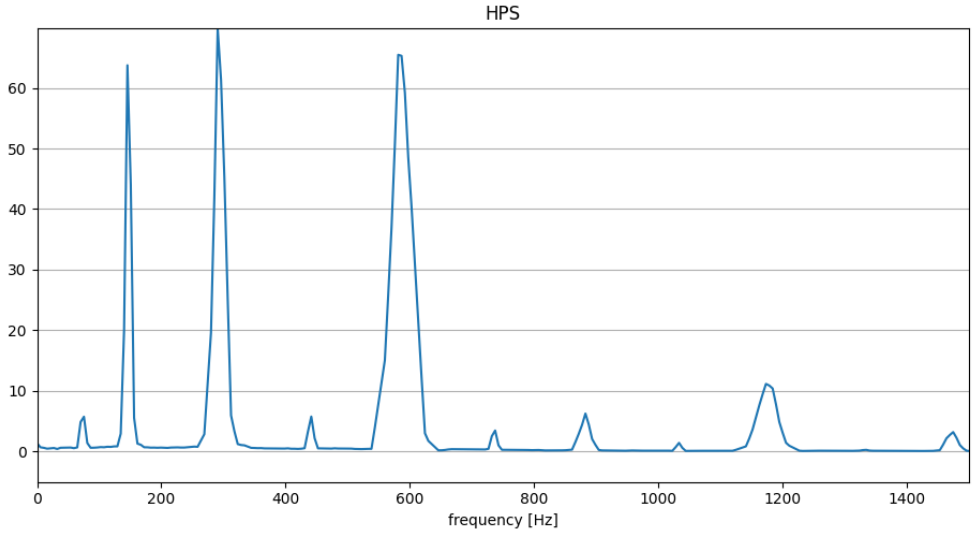
Harmonic Product Spectrum [9] polega na dodawaniu do modułu oryginalnego widma jego decymowanych wersji. Po k-krotnej decymacji modułu widma, k-ta harmoniczna przesuwa się w miejsce harmonicznej podstawowej. Po dodaniu do siebie wersji oryginalnej i decymowanej harmoniczna podstawowa ma amplitudę wyższą niż pozostałe 3.8. Operację decymacji i sumowania widma można powtórzyć kilka razy. Efektem ubocznym tej operacji jest wzrost amplitudy również artefaktów powstałych w wyniku przesunięcia samej harmonicznej podstawowej. Przy niskiej rozdzielczości widma przesunięte harmoniczne mogą się minąć, a amplituda f_0 rozkłada się na kilka prążków widma, zamiast skupić się na jednym. Rzadko zdarza się by funkcja **HPS** miała najwyższą amplitudę w miejscu faktycznej f_0. Na rysunku 3.9 harmoniczna podstawowa trafiła w najwyższą amplitudę, jednak na rysunku 3.10 już nie.



Rysunek 3.8: Rysunek przedstawia kolejne kroki metody HPS [10].



Rysunek 3.9: Na wykresie widać maksymalną wartość osi Y w okolicach 300 Hz, co odpowiada tonowi D4. Sygnał rzeczywiście jest tej częstotliwości.



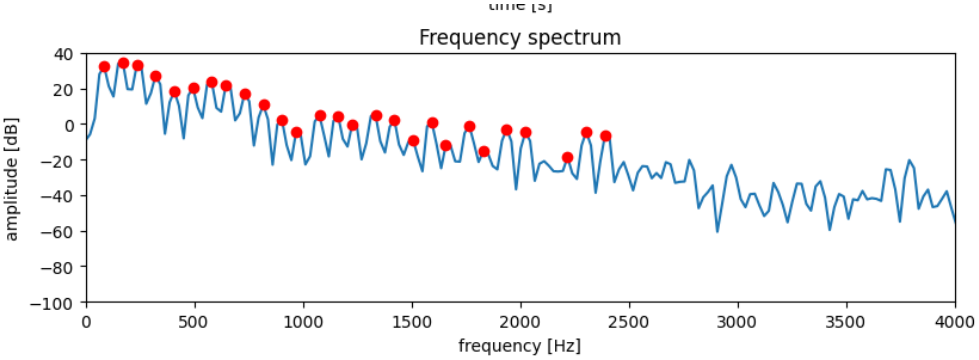
Rysunek 3.10: Na wykresie widać maksymalną wartość osi Y w okolicach 300 Hz, jednak rzeczywista częstotliwość podstawowa sygnału to 600 Hz, odpowiadająca tonowi D5.

3.3.6 Rozwiązanie autorskie, czyli FFTxAFFTxHPS

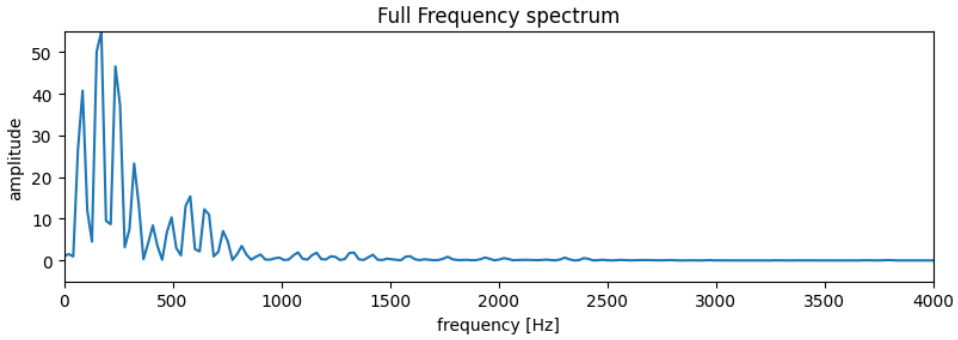
W poszukiwaniu najlepszego rozwiązania autor pracy podjął się próby stworzenia własnej medtody. *FFTxAFFTxHPS* to iloczyn wyniku *FFT*, autokorelacji widma i

funkcji *HPS* (3.8). Takie działanie ma na celu zachowanie zalet tych metod i usunięcie ich wad. Nowa metoda wytłumia artefakty funkcji *HPS* i wyróżnia oczekiwaną częstotliwość fundamentalną zachowując dokładność oryginalnego *FFT*, dzięki temu że na znaczeniu zyskują tylko wartości wspólne dla wszystkich trzech funkcji. W efekcie amplitudy pierwszych kilku harmonicznych są znacznie zwiększone, amplitudy dalszych harmonicznych pozostają bez zmian, lub są zmniejszone (rys. 3.11). Zestawienie *FFTxAFFTxHPS* razem z metodami *FFT*, *AFFT* i *HPS* można zobaczyć na rysunkach 4.1, 4.2, 4.3 i 4.4.

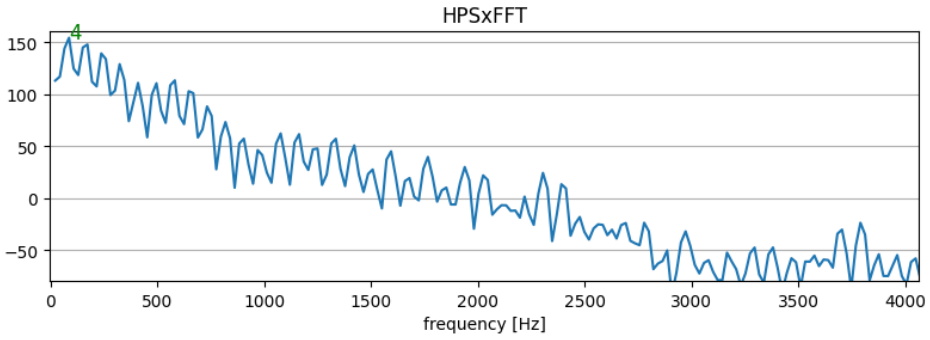
$$FFTxAFFTxHPS = FFT(x) * autocorr(FFT(x)) * HPS(x) \tag{3.8}$$



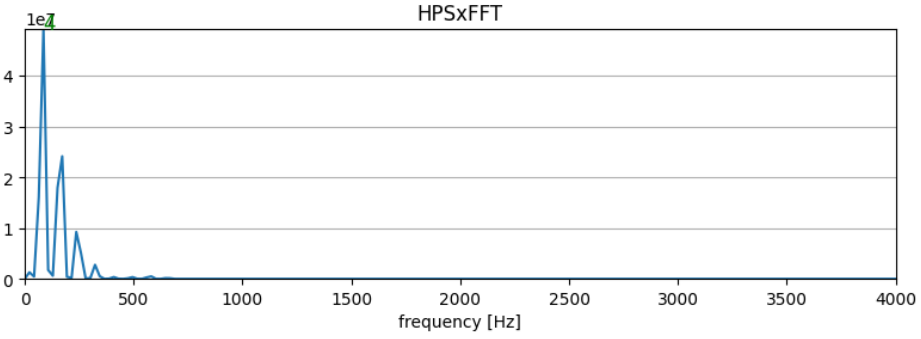
(a) Widmo w skali logarytmicznej



(b) Widmo w skali liniowej



(c) HPSxFFT w skali logarytmicznej



(d) HSPxFFT w skali liniowej

Rysunek 3.11: Powyższe obrazki przedstawiają przebiegi dźwięku E2 (ok. 82 Hz).

Rozdział 4

Implementacja aplikacji

4.1 Opis Systemu

Aplikacja służy do określenia dźwięku granego na gitarze w sposób zrozumiały dla oprogramowania syntezującego dźwięk. W przypadku aplikacji czasu rzeczywistego dźwięk jest czytany z interfejsu muzycznego, w przypadku aplikacji czasu nierzeczywistego analizowany jest sygnał zapisany w pliku. Następnie system ustala w kilku krokach jaki dźwięk jest obecny w danym fragmencie sygnału i określa częstotliwość jego harmonicznej fundamentalnej. Ostatnim etapem jest przygotowanie tej informacji w postaci czytelnej dla oprogramowania syntezującego i przekazanie mu jej za pomocą standardu MIDI.

4.2 Analiza danych i wnioski

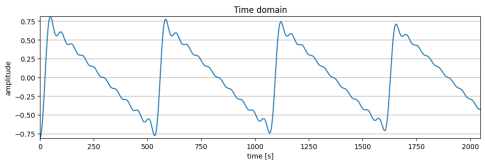
Zebrane zostały zbiory sygnałów dźwiękowych z dwóch gitar. Próbki obejmują zarówno pojedyncze dźwięki, akordy jak i sola gitarowe. W ramach badań nad charakterystyką każdego z instrumentów i ustalenia metody jej uzyskania wykorzystane zostały wszystkie nagrania, jednak aplikacja przedstawiona w tej pracy sprawnie interpretuje tylko sygnały monofoniczne, czyli pojedyncze dźwięki i sola.

4.2.1 Porównanie metod wyznaczania częstotliwości fundamentalnej

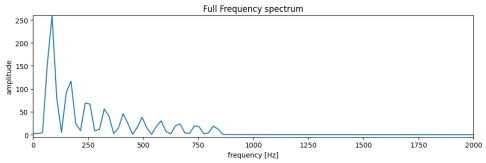
W rozdziale 3.3 opisano metody wyznaczania harmonicznej fundamentalnej i jej częstotliwości. W tej sekcji przedstawione zostanie porównanie wszystkich tych metod na kilku sygnałach:

- x1 - sygnał syntezowany, f_0 = E2 (2.406 Hz), kolejne harmoniczne są dokładnymi wielokrotnościami harmonicznej podstawowej. Amplituda f_0 jest najwyższa, amplitudy pozostałych harmonicznych opadają wraz z numerem harmonicznej (rys. 4.1).

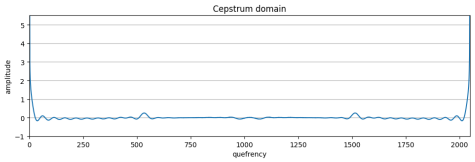
- x2 - sygnał syntezowany, f_0 i kolejne harmoniczne jak w x1. Najwyższa jest amplituda 2 harmonicznej (rys. 4.2).
- high - fragment nagrania gitary, f_0 to D5 (ok. 587 Hz). Wybrzmiewa poprzednie uderzenie D4 (ok. 294 Hz) (rys. 4.3).
- low - fragment nagrania gitary, f_0 to ok. 82 Hz. Warto zauważyć, że 2 i 3 harmoniczna mają większe energie niż podstawowa (rys. 4.4).
- Black Dog - fragment nagrania gitary, popularna sekwencja dźwięków z utwory *Black Dog* zespołu *Led Zeppelin*.



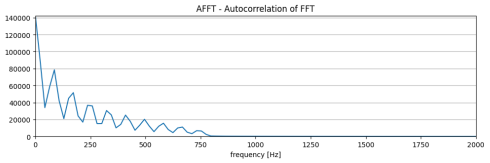
(a) Przebieg sygnału $x1$ w dziedzinie czasu.



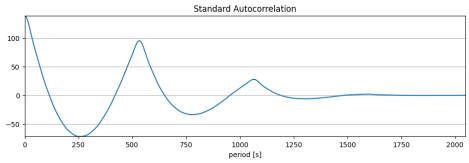
(b) Widmo w skali liniowej.



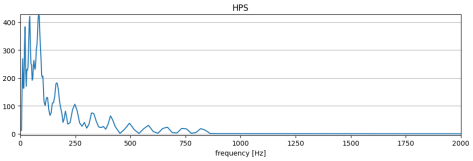
(c) Cepstrum sygnału $x1$.



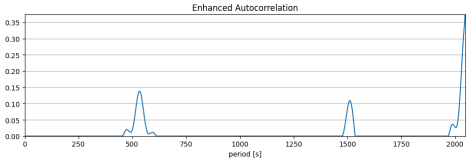
(d) Autokorelacja widma $x1$



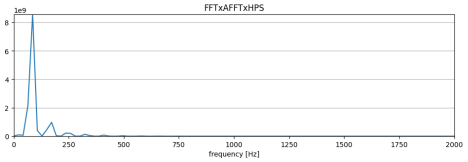
(e) Standardowa autokorelacja sygnału $x1$.



(f) Wynik funkcji HPS na $x1$

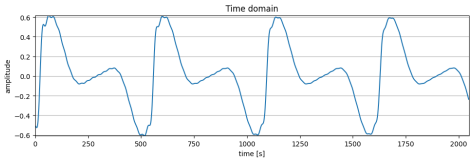


(g) Enhanced Autocorrelation sygnału $x1$.

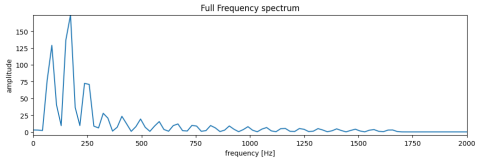


(h) Autorska funkcja FFTxAFFTxHPS

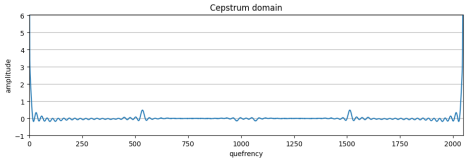
Rysunek 4.1: Powyższe wykresy przedstawiają przebiegi syntezowanego dźwięku e2 (82 Hz).



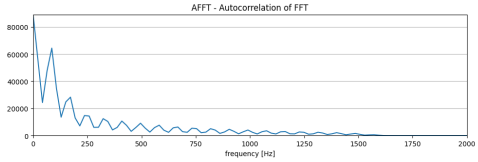
(a) Przebieg sygnału $x2$ w dziedzinie czasu.



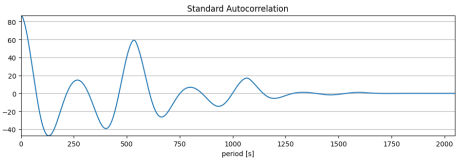
(b) Widmo w skali liniowej.



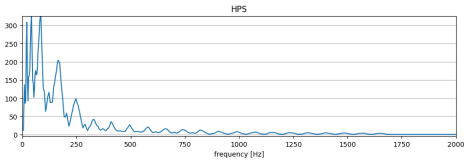
(c) Cepstrum sygnału $x2$.



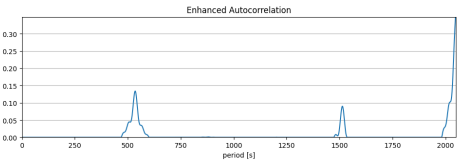
(d) Autokorelacja widma $x2$



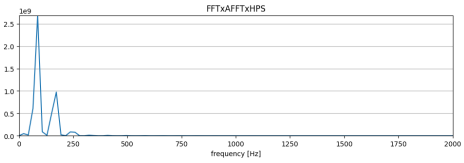
(e) Standardowa autokorelacja sygnału $x2$.



(f) Wynik funkcji HPS na $x2$

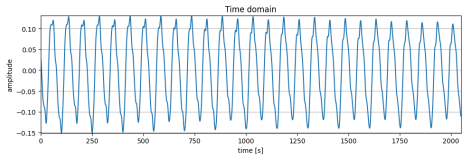


(g) EAC sygnału $x2$.

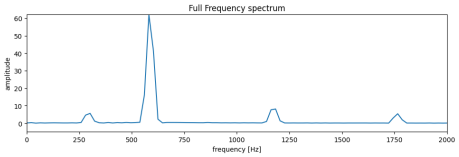


(h) Autorska funkcja FFTxAFFTxHPS

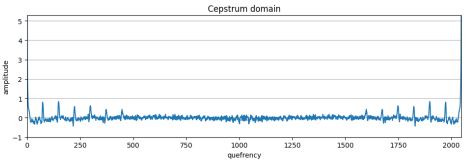
Rysunek 4.2: Powyższe wykresy przedstawiają przebiegi syntezowanego dźwięku e2 (82 Hz).



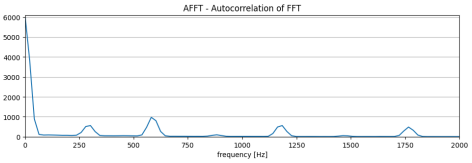
(a) Przebieg sygnału *high* w dziedzinie czasu.



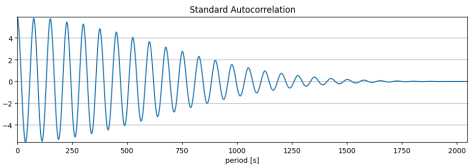
(b) Widmo w skali liniowej.



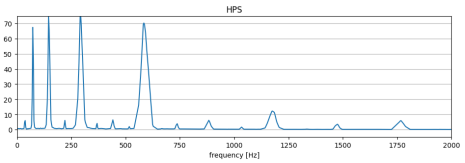
(c) Cepstrum sygnału *high*.



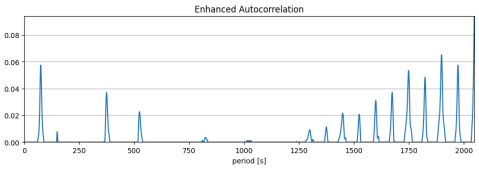
(d) Autokorelacja widma *high*



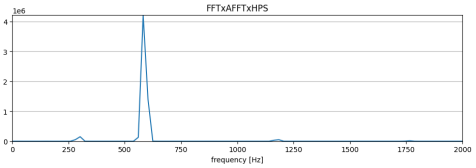
(e) Standardowa autokorelacja sygnału *high*.



(f) Wynik funkcji HPS na *high*

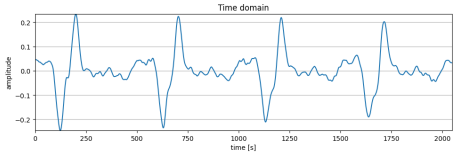


(g) EAC sygnału *high*.

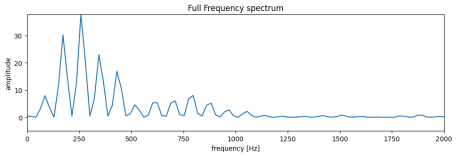


(h) Autorska funkcja FFTxAFFTxHPS

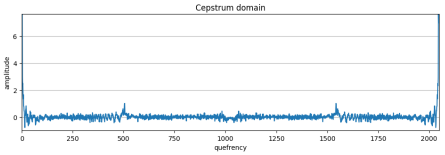
Rysunek 4.3: Powyższe wykresy przedstawiają przebiegi syntezowanego dźwięku D5 (587 Hz).



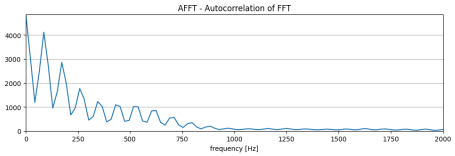
(a) Przebieg sygnału *low* w dziedzinie czasu.



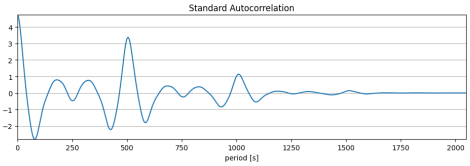
(b) Widmo w skali liniowej.



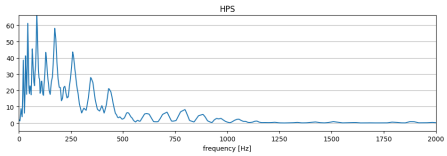
(c) Cepstrum sygnału *low*.



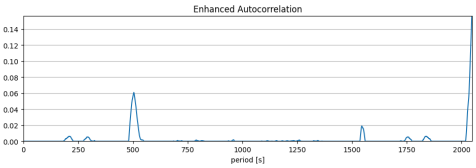
(d) Autokorelacja widma *low*



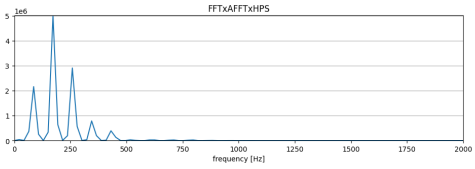
(e) Standardowa autokorelacja sygnału *high*.



(f) Wynik funkcji HPS na *low*



(g) EAC sygnału *low*.



(h) Autorska funkcja FFTxAFFTxHPS

Rysunek 4.4: Powyższe wykresy przedstawiają przebiegi syntezowanego dźwięku E2 (82 Hz).

Wyniki porównania

W powyższych zestawieniach przewagę mają metody oparte na autokorelacji - EAC, AFFT i standardowa autokorelacja sygnału w dziedzinie czasu. Wyniki o najwyższej jakości daje metoda *EAC*, ponadto jest ona najmniej kosztowna obliczeniowo ze wszystkich analizowanych rozwiązań. Metoda autorska *FFTxAFFTxHPS* niestety jest czuła na sytuacje, w których to w sygnale największą energię ma harmoniczna inna niż podstawowa.

4.2.2 Wnioski

Badania eksperymentalne wykazały, że sygnał dostarczony przez zewnętrzne karty dźwiękowe jest faktycznie bardzo wysokiej jakości. Problemy z SNR można pominąć, żadna forma wygładzania sygnału w celu usunięcia szumów nie jest potrzebna.

Barwa dźwięku

Każdy z instrumentów użytych w badaniu posiada nieco inną charakterystykę brzmienia. Te właściwości wynikają z konstrukcji, rodzaju drewna, a i jakości osprzętu elektronicznego. Barwa brzmienia ma wpływ na jakość pomiarów harmonicznej fundamentalnej metodami przedstawionymi w 3. Poprawne wyniki gwarantuje znikoma różnica amplitudy między pierwszymi kilkoma harmonicznymi, a najlepsze wyraźna przewaga f_0 nad resztą harmonicznych.

Barwa dźwięku w zależności od użytego przetwornika.

Poza unikatowym brzmieniem wynikającym z użytych materiałów, wpływ na nie ma również wybór przetwornika zbierającego sygnał. Większość gitar elektrycznych posiada dwa przetworniki elektromagnetyczne będące źródłem rejestrowanego sygnału. Przetwornik A po stronie gryfu zbiera sygnał, w którym dominują niskie harmoniczne, natomiast przetwornik B po stronie mostka rejestruje sygnał o dominujących wysokich harmonicznych. Barwa brzmienia ma wpływ na sprawność Detektora Harmonicznej Fundamentalnej. Z badań wynika, że lepsze rezultaty daje sygnał rejestrowany przez przetwornik po stronie gryfu, dzięki temu że niskie harmoniczne mają wtedy zdecydowanie bardziej zrównoważone amplitudy niż w przypadku użycia przetwornika po stronie mostka, który wyraźnie wyróżnia harmoniczne ponad f_0.

Ograniczenie pasma

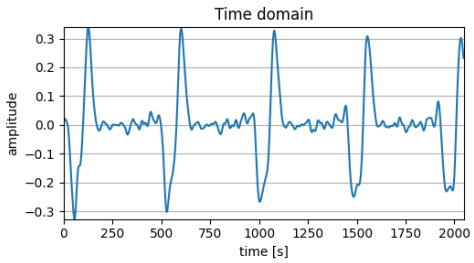
Po przebadaniu spektrum charakterystyki widmowej kilku sygnałów z obydwu gitar autor doszedł do wniosku, że informacje wartościowe dla aplikacji mieszczą się w przedziale poniżej 3.5 kHz. Ponadto wszystkie dźwięki możliwe do zagrania na gitarze 24-progowej mieszczą się w paśmie ponad dwa razy mniejszym. Dźwięk E6 (1318.5 Hz), grany na 24 progu, a nawet jego druga harmoniczna E7 (2637 Hz) również mieszczą się w tym przedziale do 3.5kHz.

Alternatywna metoda detekcji uderzenia

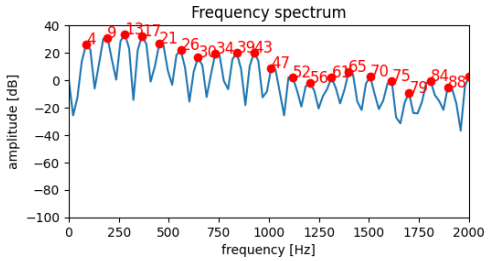
Detektor uderzenia oparty na wariancji widma i wzroście energii sygnału sprawdza się doskonale przy sygnale o wysokich częstotliwościach podstawowych. Sygnał o krótkim okresie jest gwarancją stałego przyrostu energii. Energia sygnału, którego okres mieści się w długości okna zaledwie kilka razy niestety nie jest stabilna, na skutek czego detektor uderzenia znajduje fałszywe uderzenia.

Jednym z rozwiązań tego problemu może być zmiana metody na badanie obwiedni sygnału. Krzywa obwiedni w popularnym modelu ADSR ma cztery charakterystyczne etapy **attack**, **decay**, **sustain** i **release**, wspomniane już w sekcji 3.2.

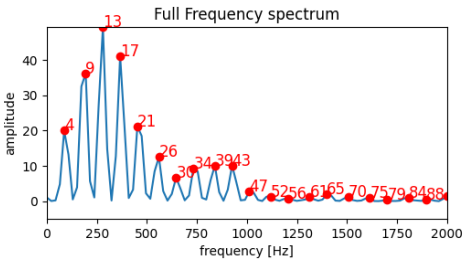
Wyróżnienie w sygnale tych czterech etapów obwiedni 3.4 pozwoliłoby w jednoznaczny i prosty sposób określić chwilę uderzenia i zakończenia wybrzmiewania dźwięku, jednak przy wyborze tej metody pojawia się kolejny nieoczywisty problem - jak określić krzywą obwiedni?



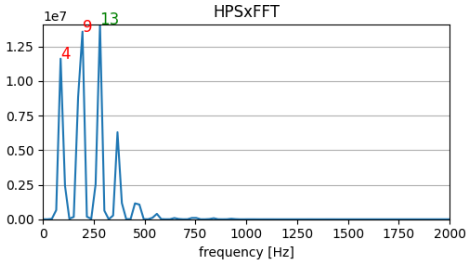
(a) Przebieg sygnału w dziedzinie czasu



(b) Widmo w skali logarytmicznej



(c) Widmo w skali liniowej



(d) Autorska funkcja HSPxFFT

Rysunek 4.5: Powyższe obrazki przedstawiają przebiegi dźwięku e2 82 Hz. Na wykresach widma widać, że trzecia harmoniczna dominuje nad f_0 i f_1. Na wykresie 4.5d zaznaczona jest korekcja pomiaru - znalezione zostały szczyty 4 i 9.

Korekcja FFTxAFFTxHPS

W rozdziale 3.3 opisane zostały metody określania częstotliwości fundamentalnej, a także przewidziane ich wady i zalety. Autor przedstawił własną metodę *FFTxAFFTxHPS*, postulując że jest niezawodna i lepsza zarówno od funkcji autokorelacji widma i funkcji HPS. W rzeczywistości okazało się, że jej niezawodność jest wysoka dla średnich i wysokich dźwięków - poniżej pewnego tonu wynoszącego około 147 Hz, D3 - jakość tej metryki drastycznie spada. Błąd wynika z charakterystyki widma niskich dźwięków. Amplituda drugiej, a nawet trzeciej harmonicznej na ogół jest wielokrotnie większa niż amplituda f_0. W takim wypadku przemnożenie widma przez wynik funkcji HPS, ani nawet funkcji autokorelacji nie jest w stanie wybić f_0 ponad pozostałe dominujące harmoniczne. Ten stan rzeczy można częściowo skorygować znajdując szczyty przed maximum *FFTxAFFTxHPS* i poddać je dalszej analizie w celu ustalenia prawdziwej f_0. Przykład widma niskiego tonu i odpowiadającej mu metryki *FFTxAFFTxHPS* oraz jej korekcję widać na rysunku 4.5.

4.3 Implementacja aplikacji

4.3.1 Technologie

Język Programowania

Autor przy wyborze języka programowania, w którym została napisana aplikacja kierował się kilkoma kryteriami:

- dostępność języka oprogramowania,
- dostępność bibliotek dotyczących zagadnienia przetwarzania sygnału,
- znajomość języka.

Pierwsza część badań została przeprowadzona w środowisku Matlab. Po ich zakończeniu, do puli kandydatów dołączono język Python ze względu na jego open-source'owy charakter i mnogość sprawdzonych już bibliotek. W szczególności biblioteki **numpy** [11] i **matplotlib** [12], które imitują działania na macierzach i tworzenie wykresów w sposób znany ze środowiska Matlab, przeważyły o wyborze języka Python do implementacji systemu.

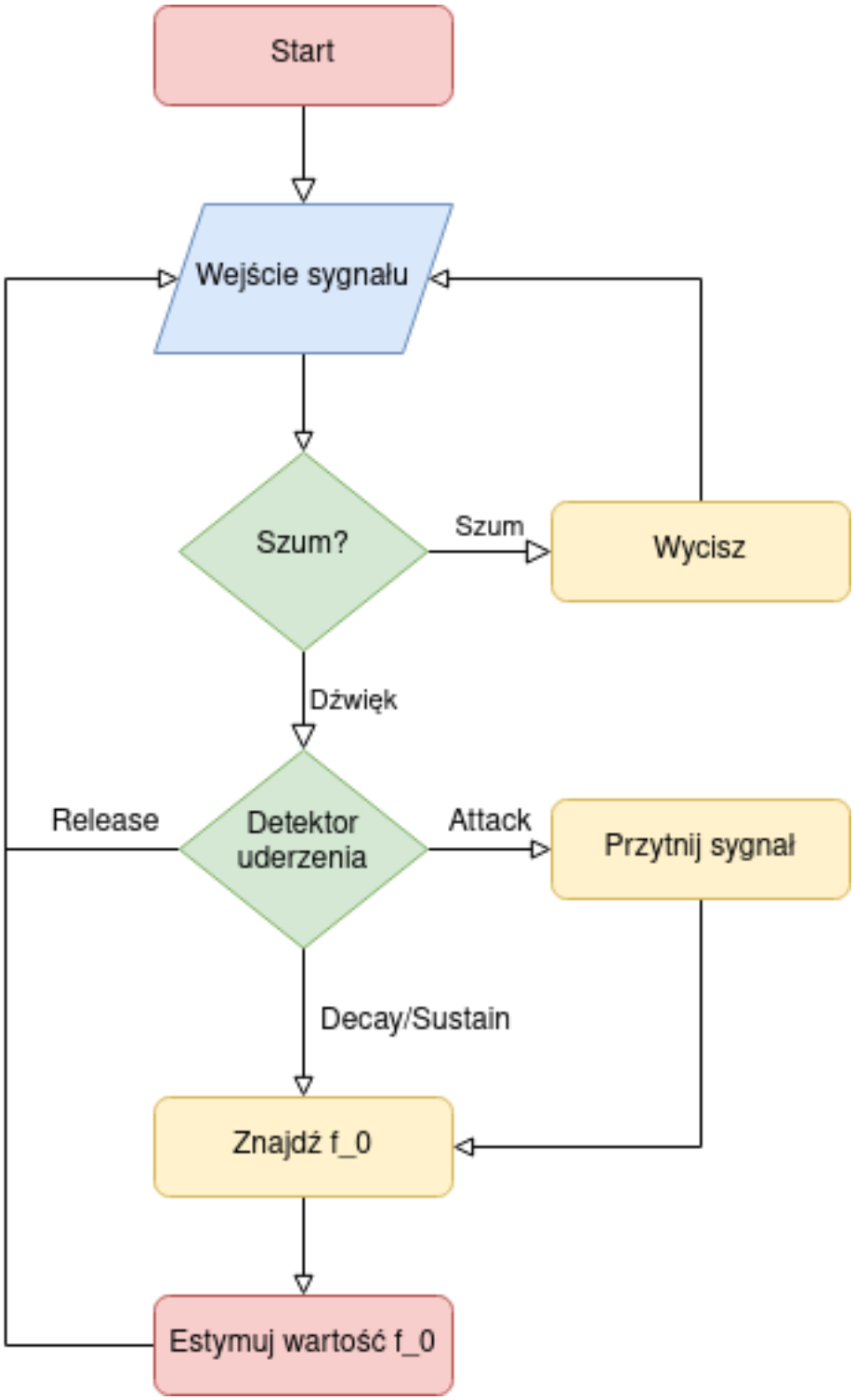
Zewnętrzna karta dźwiękowa

Karta dźwiękowa zapewnia wysokiej jakości dźwięk w czasie rzeczywistym. Popularne urządzenia tego typu mają bardzo niskie poziomy szumu i pozwalają na nagrywanie dźwięku z próbkowaniem od 44.1 kHz nawet do 192kHz. Autor wykorzystał karty Focusrite Scarlet 2i4 [13], a także Roland Quad-Capture 2x2 [14]. W tej pracy używana była częstotliwość 44.1 kHz.

Syntezator Dźwięku

Urządzeniem, które przyjmuje wiadomości MIDI wysyłane przez system, następnie generuje dźwięk na ich podstawie, jest syntezator dźwięku. Autor zaleca użycie syntezatora w postaci oprogramowania komputerowego.
Na systemie operacyjnym Linux popularnym i darmowym oprogramowaniem tego typu jest Yoshimi [15], na systemie Windows - Virtual Midi Synth [16].

4.3.2 Schemat blokowy aplikacji



4.3.3 Podział systemu na moduły

System został podzielony na kilka modułów:

- rejestrator dźwięku 4.3.3,
- bramka szumów,
- detektor uderzenia,
- detektor harmonicznej fundamentalnej,
- estymator częstotliwości fundamentalnej,
- konwerter częstotliwości do notacji w standardzie MIDI

W kolejnych sekcjach każdy z modułów zostanie opisany.

Rejestrator dźwięku

Dźwięk jest rejestrowany za pomocą zewnętrznej karty dźwiękowej z częstotliwością 44.1 kHz. Zadaniem modułu jest pobranie nowych 256 próbek gdy tylko będą dostępne, następnie dodanie ich do wektora przechowującego sygnał w dziedzinie czasu o długości 2048 próbek i uruchomienie Bramki Szumów. Taki mechanizm przesuwania **okna** 2048 próbek o krok 256 próbek zapewnia rozdzielczość widma na poziomie 22.5 Hz i możliwość reakcji na dynamiczne zmiany. Wartości 2048 i 256 są wspomniane w artykule [17], według autorów zapewniają najlepsze wrażenia użytkownikowi. Pełne okno 2048 próbek przed przejściem do następnego kroku jest przemnażana przez okno Hamminga.

Bramka szumów

Do analizy nadaje się tylko sygnał o odpowiedniej energii i właściwościach harmonicznych. O tym czy będzie przekazany do kolejnych modułów decyduje Bramka Szumu na podstawie energii 2048 próbek sygnału. Przyjęto proste kryterium, które przepuszcza sygnał o mocy co najmniej 0.03.

Detektor uderzenia

Rozróżnienie pojedynczych uderzeń jest niezbędne, aby naturalnie symulować następujące po sobie dźwięki. Poprzestanie na znalezieniu częstotliwości i bramce szumów może skutkować zlaniem kilku uderzeń o tym samym tonie w jeden długi dźwięk. Detekcja uderzenia opiera się na obserwacji wariancji widma 3.2.2 i zmianie pochodnej energii sygnału 3.2.1. Jeżeli energia ostatnich trzech ramek sygnału ma tendencję rosnącą, a do tego jej średnia pochodna ma wartość większą niż empirycznie wyznaczone 5, to w analizowanym sygnale nastąpiło uderzenie.

Detektor częstotliwości fundamentalnej

Detekcja Harmonicznej Fundamentalnej opiera się na jednej z dwóch metod analizy sygnału opisanych w sekcji 3.3 - *FFTxAFFTxHPSv2* i *EAC*. Wybrana funkcja zwraca szczyt f_0, następnie znaleziony zostaje szereg harmonicznych z nim związanych. Te dane są przekazane do Estymatora częstotliwości fundamentalnej.

Do ustalenia szeregu będą potrzebne szczyty widma, które zwraca funkcja *find_peaks* z pakietu Python *scipy* [18]. Amplitudy widma są przedstawione w skali logarytmicznej. Parametry *find_peaks* to:

- minimalna wysokość szczytu,
- wysokość względna szczytu,
- minimalny odstęp między szczytami

Autor ustalił że najlepsze wyniki daje minimalna wysokość szczytu równa -20dB i wysokość względna szczytu równa 5dB. Odstęp został pominięty. Następnym krokiem jest ustalenie szeregu rozpoczynającego się od prążka f_0 ze zbioru szczytów zwróconych przez *find_peaks* i dokładna estymacja częstotliwości fundamentalnej jak w 3.1.2.

Tablica 4.1: Implementacja EAC.

```
import numpy as np

def eac(tbuffer, factors=[2, 3]):
    """Implementation of simplified Tero Tolonen EAC algorithm"""

    def resample(x, factor, kind='linear'):
        n = np.ceil(x.size / factor)
        f = interp1d(np.linspace(0, 1, x.size), x, kind)
        return f(np.linspace(0, 1, int(n)))

    # spectral multiplication == time convolution
    fxcorr = np.fft.ifft( np.power(np.abs(np.fft.fft(tbuffer)), 0.67 ))

    # clip all values below zero to zero
    fxcorr[fxcorr < 0] = 0

    for r in factors:
        # time-scale
        fxsub = resample(fxcorr, 1/r)
        fxsub = fxsub[:tbuffer.size]

        # subtract time-scaled from original autocorrelation
        fxcorr -= fxsub

        # clip again
        fxcorr[fxcorr < 0] = 0

    # find index of maximum value
    fmax = np.argmax(fxcorr[:tbuffer.size//2])

    return fxcorr, fmax
```

Tablica 4.2: Metoda FFTxAFFTxHPS przed korekcją.

```
def fft_afft_hps(fbuffer, hps_len, win_len, afft_vec,
                 hps_vec=None):
    fft_vec = np.abs(fbuffer[:win_len//2])

    hps_fft_xcorr_vec = fft_vec * afft_vec * hps_vec

    # find firt raise
    first_val_raise = np.argmax(np.greater(hps_fft_xcorr_vec[1:],
                                           hps_fft_xcorr_vec[:-1]))
    hps_fft_xcorr_vec[:first_val_raise + 1] = 0t
    hps_fft_xcorr_max = np.argmax(hps_fft_xcorr_vec)

    return hps_fft_xcorr_max, hps_fft_xcorr_vec
```

Estymator częstotliwości fundamentalnej

Widmo częstotliwościowe przy próbkowaniu 44.1 kHz i długości ramki 2048 ma rozdzielczość zaledwie 22.5 Hz. Aby wyniki były bliższe rzeczywistości należy podnieść tę rozdzielczość. Można ten efekt uzyskać interpolując widmo. Dokładność należy zwiększyć jeszcze bardziej szukając największego wspólnego dzielnika interpolowanego już szeregu harmonicznych, tak jak to opisano w 3.1.2.

Konwerter częstotliwości do notacji w standardzie MIDI

Konwerter korzysta z tabeli w której do tonów w zakresie A0 - E6 przyporządkowane są ich częstotliwości i odpowiadające numery nut w standardzie MIDI. Tabela jest podobna do tej z [19]. Po znalezieniu nuty MIDI najbliższej f_0 wysyłana jest wiadomość MIDI do urządzenia określonego w konfiguracji początkowej aplikacji. Po zakończeniu tego etapu cały cykl systemu jest powtarzany od początku.

Poniżej (rys. 4.3) przedstawiono implementację modułu *midi* wykonującego opisane wyżej kroki. Podczas uruchomienia aplikacji otwierany jest kanał midi na wskazanym porcie (**midi_port**), następnie tworzony jest słownik *midi_dict* wykorzystywany przy transkrypcji częstotliwości przez funkcję *freq_to_midi*. Funkcja *play_midi* odpowiada za wysłanie wiadomości MIDI.

Tablica 4.3: Moduł midi.py obsługujący operacje związane z połączeniem MIDI i transkrypcją f_0 do MIDI.

```
import sys
import rtmidi


class Midi:
    """Handles operations related to MIDI connection and final note
    transcription."""
    def __init__(self, midi_port):
        self.midi_port = midi_port
        self.midi_init()
        self.freq_to_midi_init()
        self.played_midi_notes = []

    def midi_init(self):
        """Connects software to midi device on [midi_port] channel."""
        self.midiout = rtmidi.MidiOut()

        if self.midiout.get_ports():
            self.midiout.open_port(self.midi_port)
        else:
            log.error(f'rtmidi: MIDI port {midi_port} does not exist!')
            sys.exit(0)

    def freq_to_midi_init(self):
        """Initiates an array with midi notes and frequency bindings."""
        names = ['A', 'A#', 'B', 'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G',
     'G#']

        # format:
        #   {frequency[Hz] :
        #       ["Note Name", midi_note, number_of_appearances]}
        self.midi_dict = {np.power(2, i/12)*27.50 : [f'{names[i % 12]}{(i +
     9) // 12}', 21 + i, 0] for i in range(68)}

    def freq_to_midi(self, freq):
        """Converts frequency [Hz] to midi note number."""

        midi_key_freq = min(self.midi_dict.keys(), key = lambda
    midi_key_freq: abs(midi_key_freq - freq))

        return self.midi_dict[midi_key_freq][1]

    def play_midi(self, midi_note):
        """Plays a new note only if the note isn't already being played"""
        if midi_note not in self.played_midi_notes and len(self.
    played_midi_notes) < 2:
            self.stop_midi()
            try:
                self.midiout.send_message([0x90, midi_note, 100])
                self.played_midi_notes.append(midi_note)
                log.info(f'midi_note_on = {midi_note}')
            except TypeError as e:
                log.error("Midi Note has to be integer")
```

4.4 Analiza otrzymanych wyników

W tej sekcji przedstawiono końcowe wyniki i wnioski dotyczące systemu analizy sygnału muzycznego. Analizowane pliki muzyczne są tymi samymi, które wykorzystano w badaniach (sekcja 4.2.1). Dla każdego sygnału łańcuch modułów jest niezmienny, poza modułem Detektora częstotliwości fundamentalnej, w którym ponownie zostały wykorzystane wszystkie opisane metody. Na rysunkach 4.6, 4.7, 4.8 i 4.9 przedstawiono spektrogramy o długości 3 sekund, następnie czerwoną linią zaznaczono na nich rozpoznane częstotliwości f_0. Skala częstotliwości została ograniczona do 2000 Hz. Należy zauważyć, że są to już wartości obowiązujące w stroju równomiernie temperowanym, który to dzieli oktawę na 12 równych części. Zastosowano dwa warianty metody *FFTxAFFTxHPS*, oryginalny i zmodyfikowany zgodnie z opisem (4.2.2).

4.4.1 Końcowe wyniki

Detekcja f_0

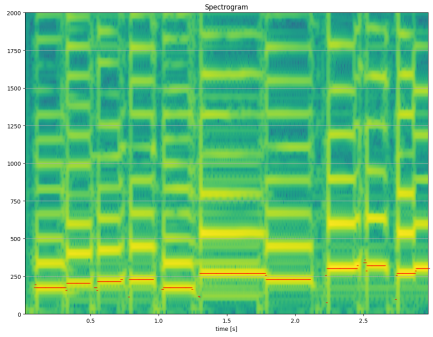
Analizowane sygnały są skrajnie różne, tak aby możliwie najlepiej odzwierciedlały rzeczywiste zastosowanie aplikacji. Po wstępnej analizie odrzucone zostały metody HPS i FFTxAFFTxHPSv1 (FFTxAFFTxHPS w oryginalnej wersji). Metoda HPS nie trafiła w f_0 ani razu. Metoda FFTxAFFTxHPSv1 tak jak przewidziano nie radzi sobie z niskimi dźwiękami, co widać na rys. 4.9.

Pozostałe metody dają wyniki podobnej jakości. Metoda AFFT została odrzucona ze względu na to, że jej dokładność może być poprawiona niewielkim kosztem, czego efektem jest funkcja *FFTxAFFTxHPSv2* (po korekcji). Standardowa autokorelacja również została odrzucona - funkcja EAC jest oparta na mniej kosztownym mnożeniu w dziedzinie częstotliwości i daje lepsze wyniki.

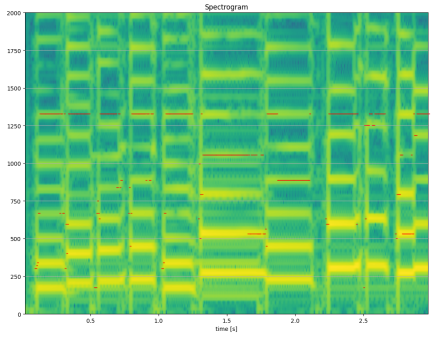
Wyróżnione zostały metody *EAC* i autorska metoda *FFTxAFFTxHPSv2*. EAC jest mniej kosztowne obliczeniowo, jednak to FFTxAFFTxHPSv2 jest bardziej niezawodne. Na rysunku 4.7 widać błędny pomiar EAC pod koniec wybrzmiewania sygnału. Autorska jest w tym przypadku bezbłędna.

Detekcja uderzenia

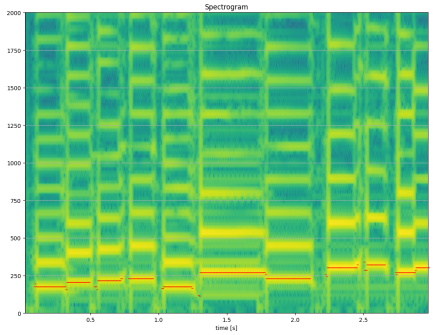
Na każdym wykresie tuż przed stabilizacją sygnału widać błędne pomiary f_0. Przyczyną takiego zachowania jest prawdopodobnie nadczuły detektor uderzenia, wykrywający przyrost energii jeszcze zanim sygnał jest ustabilizowany i wypełnia większość badanego okna. Ten problem pojawia się znacznie częściej przy rozpoczęciu uderzenia, niż przy jego zakończeniu. Niskie dźwięki zamknięte w krótkim oknie charakteryzują się brakiem stabilności energii sygnału, co skutkuje znajdowaniem przez detektor fałszywych uderzeń, czyli przyrostów energii kwalifikowanych do nowego uderzenia w obrębie tego samego uderzenia.



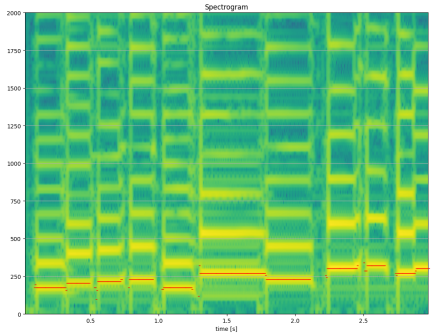
(a) Przebieg AFFT.



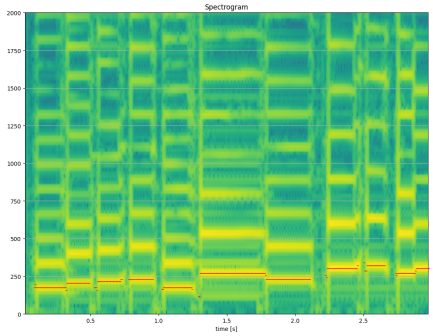
(b) HPS.



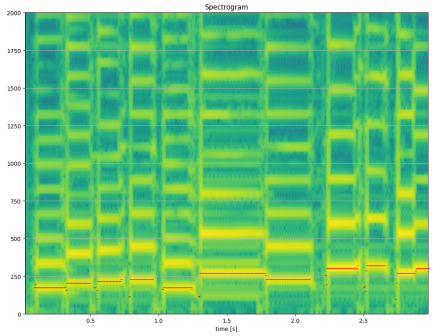
(c) Standardowa autokorelacja.



(d) Autorska funkcja FFTxAFFTxHPS przed korekcją.

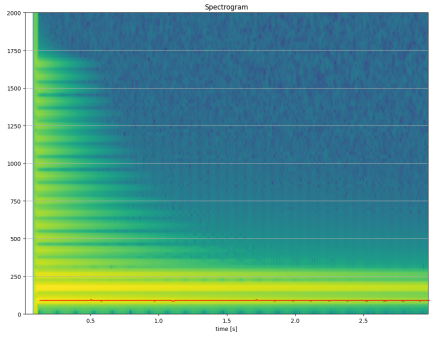


(e) EAC.

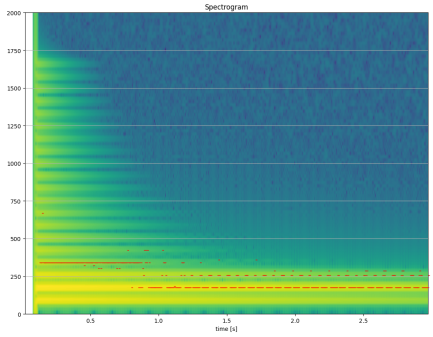


(f) Autorska funkcja FFTxAFFTxHPS.

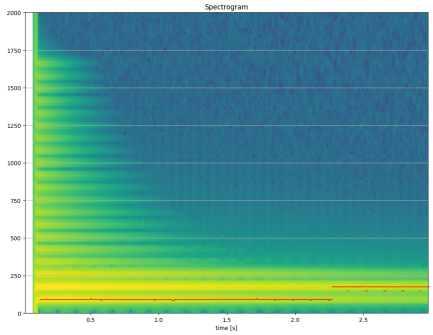
Rysunek 4.6: Powyższe obrazki przedstawiają przebiegi utowru *Black Dog*.



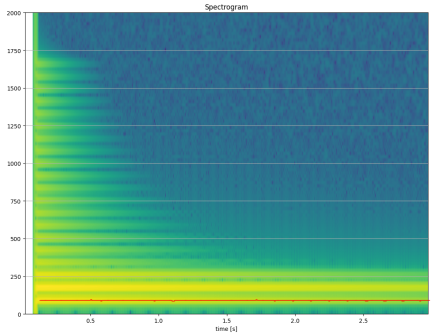
(a) Przebieg AFFT.



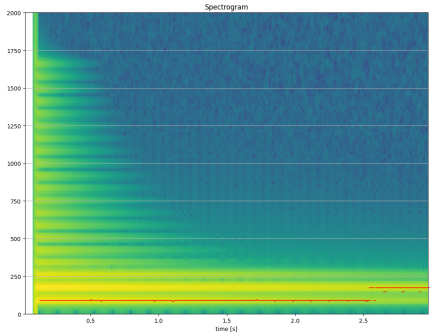
(b) HPS.



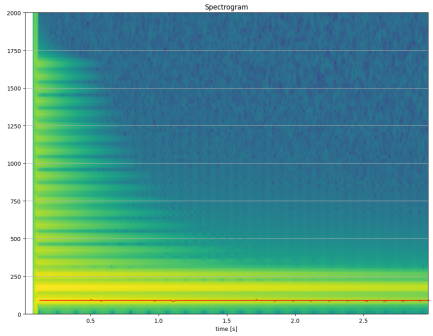
(c) Standardowa autokorelacja.



(d) Autorska funkcja FFTxAFFTxHPS przed korekcją.

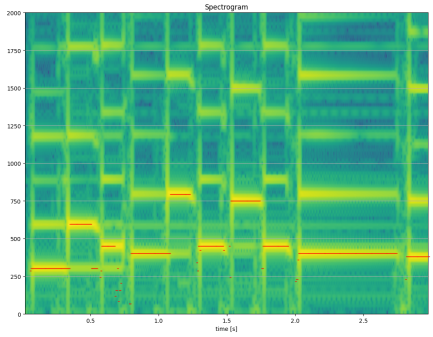


(e) EAC.

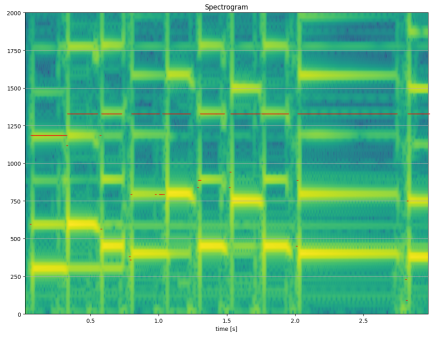


(f) Autorska funkcja FFTxAFFTxHPS.

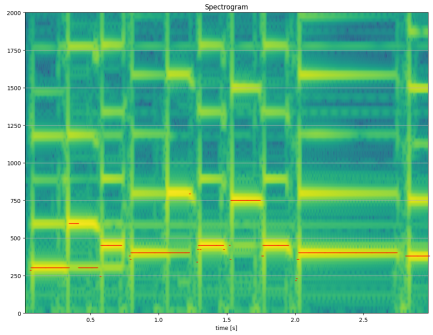
Rysunek 4.7: Powyższe obrazki przedstawiają przebiegi sygnału *x2*.



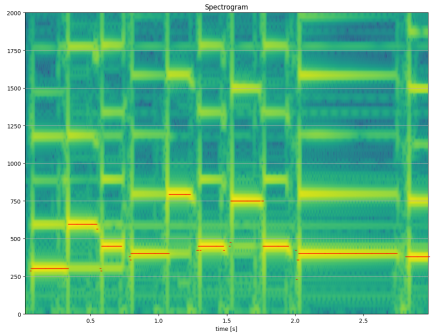
(a) Przebieg AFFT.



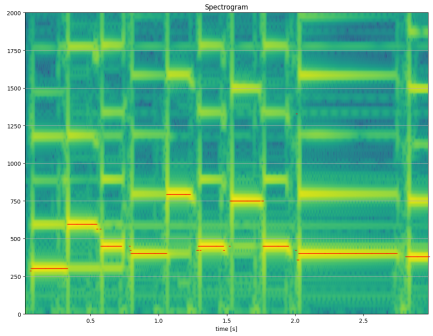
(b) HPS.



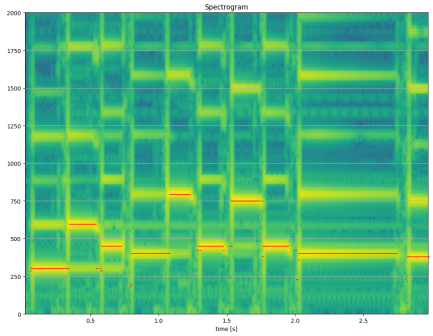
(c) Standardowa autokorelacja.



(d) Autorska funkcja FFTxAFFTxHPS przed korekcją.

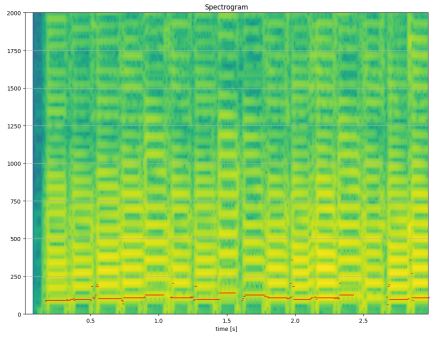


(e) EAC.

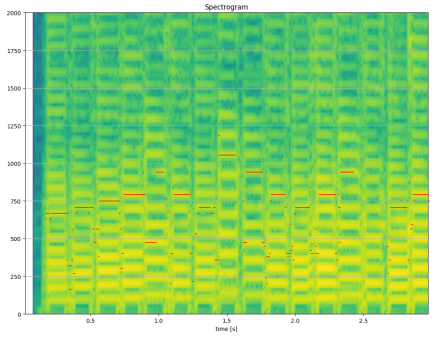


(f) Autorska funkcja FFTxAFFTxHPS.

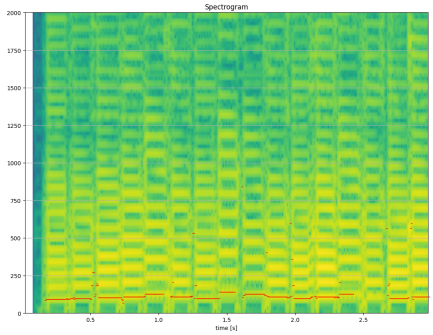
Rysunek 4.8: Powyższe obrazki przedstawiają przebiegi nagrania *high*.



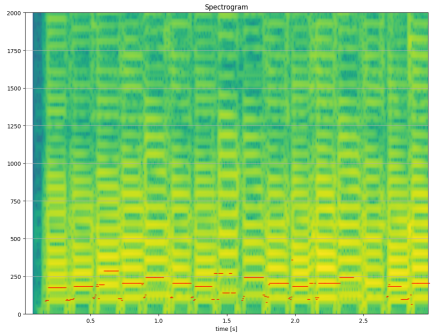
(a) Przebieg AFFT.



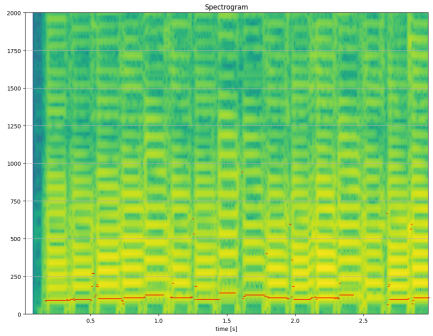
(b) HPS.



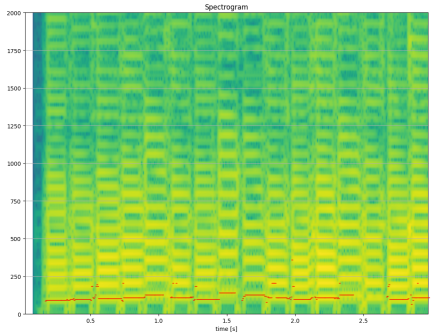
(c) Standardowa autokorelacja.



(d) Autorska funkcja FFTxAFFTxHPS przed korekcją.



(e) EAC.



(f) Autorska funkcja FFTxAFFTxHPS.

Rysunek 4.9: Powyższe obrazki przedstawiają przebiegi nagrania *low*.

Podsumowanie

Zgodnie z założeniami autor zaprojektował i zaimplementował aplikacje do rozpoznawania dźwięku i transkrypcji do notacji MIDI spełniającą ustalone kryteria. Stworzono bazę nagrań gitary, które następnie zostały kompleksowo przeanalizowane. W wyniku badań wyróżniono najbardziej obiecujące algorytmy analizy sygnału. Autor podjął próbę stworzenia własnej metody określania tonu podstawowego sygnału *FFTxAFFTxHPS*, która razem z metodą *EAC* dają najlepsze wyniki.
Wybrane algorytmy analizy sygnału muzycznego w niezawodny sposób pozwalają przenieść sygnał analogowego instrumentu do dziedziny cyfrowej. Demo aplikacji [20] prezentuje przetwarzanie dźwięku w czasie rzeczywistym, z analogowego do opisu MIDI i syntezę za pomocą różnych modułów brzmieniowych.

Dalszy rozwój

Pomimo osiągnięcia sukcesu w spełnieniu podstawowych założeń przy tworzeniu projektu opisanego w tej pracy, wiele problemów nie jest jeszcze rozwiązanych. Poniżej przedstawiono kilka najważniejszych tematów, które można w przyszłości rozwinąć.

Detektor uderzenia

Moduł określający częstotliwość fundamentalna i moduł odpowiedzialny za komunikację z urządzeniami MIDI są faktycznie sprawne i niezawodne. Problemy sprawia detektor uderzenia, który jest w formie opisanej w tej pracy zbyt czuły na zmiany energii. Dobrym pomysłem jest rozwinięcie go o nowe metody analizy obwiedni sygnału w celu poprawienia jakości pomiarów.

Detektor Amplitudy f_0

W obecnej konfiguracji wszystkie dźwięki mają tę samą amplitudę. Algorytm *BY1* może zwracać informację o amplitudzie analizowanej częstotliwości i warto wykorzystać ją przy dalszym rozwoju projektu. Dokładniejsze odzwierciedlenie amplitudy pozytywnie wpłynie na dynamikę i naturalność brzmienia syntezowanego sygnału.

Detektor Odchylenia od f_0

Jedną z podstawowych technik gitarowych jest *bend* - czyli naciągnięcie struny, dające efekt podwyższenia dźwięku. W tej pracy rozpoznawanie bendów nie zostało

zaimplementowane. Aby określić wysokość podciągnięcia wystarczy wiedza o stanie uderzenia (attach, decay, sustain) i odchylenie aktualnego dźwięku od f_0. Jeżeli nastąpiło odchylenie od pierwotnego f_0 w obrębie jednego uderzenia, to jest to podciągnięcie.

Detektor Slide

Slide to kolejna popularna i podstawowa technika gitarowa, polega na zmianie wysokości dźwięku poprzez przesunięcie palca z jednego progu na inny, na tej samej strunie. Slide nie jest tak płynny w zmienia wysokości dźwięku jak podciągnięcie. Możliwe, że przeskok palca między progami jest podobny do uderzenia o niskiej amplitudzie, co może sprawiać problem przy detekcji tej techniki.

System polifoniczny

System opisany w tej pracy jest systemem monofonicznym - rozpoznaje tylko jeden dźwięk w danej chwili. Kolejnym krokiem do przodu byłoby stworzenie systemu rozpoznającego dwudźwięki lub nawet akordy. Większość modułów stworzonych na potrzeby tej pracy można by wykorzystać ponownie w systemie polifonicznym.

Bibliografia

[1] PWN. *transkrypcja*. URL: https://encyklopedia.pwn.pl/haslo/transkrypcja;3988755.html.

[2] Jam Origin. *MIDI Guitar 2*. URL: https://www.jamorigin.com/.

[3] MiGiC. *MIDI Guitar 2*. URL: https://migic.com/.

[4] Jan Schlüter. *Deep Learningfor Event Detection,Sequence Labelling andSimilarity Estimationin Music Signals*. JOHANNES KEPLERUNIVERSITY LINZ, 2017.

[5] M. Bertocco, C. Offelli i D. Petri. "Analysis of damped sinusoidal signals via a frequency-domain interpolation algorithm". W: (1994).

[6] Tobias R. *ADSR*. URL: https://commons.wikimedia.org/w/index.php?curid=1237705.

[7] *Autokorelacja*. URL: https://sound.eti.pg.gda.pl/~greg/dsp/04-Splot.html.

[8] Tero Tolonen i Matti Karjalainen. "A Computationally Efficient MultipitchAnalysis Model". W: (2000).

[9] Manfred R. Schroeder. *Period histogram and product spectrum: new methods for fundamental-frequency measurement*. 1968.

[10] *Tempo and beat tracking for audio signals with music genre classification*. URL: https://www.researchgate.net/publication/220194731_Tempo_and_beat_tracking_for_audio_signals_with_music_genre_classification.

[11] *numpy documentation*. URL: https://numpy.org/doc/1.19/.

[12] *matplotlib documentation*. URL: https://matplotlib.org/3.3.3/contents.html.

[13] Focusrite. *Scarlett 2i4*. URL: https://focusrite.com/en/usb-audio-interface/scarlett/scarlett-2i2.

[14] Roland. *Roland Quad-Capture 2x2*. URL: https://www.roland.com/global/products/quad-capture/.

[15] *Yoshimi*. URL: http://yoshimi.sourceforge.net/.

[16] *Virtual Midi Synth*. URL: https://coolsoft.altervista.org/en/virtualmidisynth.

[17] Rodrigo Schramm i in. "A polyphonic pitch tracking embedded system for rapid instrument augmentation". W: (2018).

[18] *scipy findpeaks documentation*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html.

[19] *Midi notes*. URL: https://www.inspiredacoustics.com/en/MIDI_note_numbers_and_center_frequencies.

[20] Antoni Jankowski. *Repozytorium*. URL: https://gitlab.com/anthonio/pythonaudioanalyserdemo.